BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

Módulos para la Transferencia Segura y Secreta de Información

María Eugenia Ansalas

UNLP - Mayo 1998

TES
98/1
DIF-02004
SALA

UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMATICA
Biblioteca
50 y 120 La Plata
catalogo.info.unlp.edu.ar
biblioteca@info.unlp.edu.ar

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

MODULOS PARA LA TRANSFERENCIA SEGURA Y SECRETA DE INFORMACION

TRABAJO DE GRADO

Alumno : María Eugenia Ansalas

Director : Horacio Villagarcia Wanza

MAYO 1998

Facultad de Ciencias Exactas
UNLP

INDICE

Páginas

OBJETIVO

Los requisitos para mantener la información en forma secreta y segura dentro de una organización han sufrido dos importantes cambios en estas últimas décadas. Con la introducción de las computadoras, la necesidad de proveer herramientas automáticas para la protección de archivos y otra información almacenada se hacia cada vez más notoria. Tal es el caso de los sistemas compartidos, siendo más crítico para los sistemas que pueden ser accedidos a través de un teléfono público o una red de datos.

El segundo cambio importante que afectó la seguridad es la introducción de sistemas distribuidos y el uso de las facilidades otorgadas por las redes y las comunicaciones para transmitir datos entre usuarios (terminales) y computadoras, y entre computadoras. Las medidas de seguridad en una red son necesarias para proteger los datos durante su transmisión.

En un momento en el que la seguridad en la transmisión de información alcanza niveles de gran interés y con el objeto de difundir y proporcionar herramientas que permitan el análisis de los distintos métodos criptográficos, he diseñado y desarrollado un ambiente de prueba compuesto por módulos que ilustran la transmisión de información en forma secreta y segura entre usuarios "remotos" del sistema y permite realizar "transmisiones de mensajes" haciendo uso de métodos criptográficos.

Se consideraron para el presente trabajo los algoritmos de encriptación simétrica y asimétrica estándares y un método de sustitución simple con el objeto de poder realizar comparaciones entre los distintos métodos. Los algoritmos seleccionados (DES y RSA) son los más utilizados y han resultado ser los más confiables hasta el momento.

Asimismo, otro de los objetivos del presente trabajo es proporcionar un ambiente donde se adopte el rol de criptoanalista, analizando la pérdida de información o modificación parcial de los mensajes originales frente a los ataques y/o distorsiones efectuadas por eventuales atacantes durante la transmisión de información en un ambiente multiusuario.

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

I.- INTRODUCCION

Para medir efectivamente el grado de seguridad en una organización y para evaluar y elegir productos y políticas de seguridad, consideremos tres aspectos de la seguridad de la información:

- **Ataques**: cualquier acción que comprometa la seguridad de la información propia de una organización.
- **Mecanismos de seguridad**: son diseñados para detectar, prevenir y recuperarse de un ataque.
- **Servicios de seguridad**: mejoran la seguridad de un sistema de procesamiento de datos y la transferencia de información en una organización. El servicio de seguridad se opone a los ataques y hace uso de uno o más mecanismos de seguridad para proveer el servicio.

Podemos pensar a los ***servicios de seguridad*** de la información como copia del tipo de funciones normalmente asociadas con los documentos físicos. Gran parte de las actividades de la humanidad, en áreas tan diversas como comercio, política exterior o acciones militares, dependen del uso de documentos y de la confianza que tengan ambas partes en la integridad de éstos. Al adquirir los sistemas de información mayor importancia, la información electrónica ha reemplazado en muchos casos al documento en papel, por lo que las funciones que se realizaban sobre los documentos físicos para proveer la seguridad deben realizarse de alguna manera sobre su forma electrónica. Una clasificación usual de los servicios de seguridad es la siguiente:

- *Confidencialidad* : Requiere que la información en un sistema de computación y transmisión de información sea accesible solo para lectura para las partes autorizadas. Este tipo de acceso incluye impresión, visualización y otras formas de acceso, incluyendo la simple verificación de la existencia de un objeto.
- *Autenticación* : Requiere que el origen de un mensaje sea correctamente identificado, con la certificación de que la identificación no es falsa.

- *Integridad* : Requiere que los componentes del sistema de computación y la transmisión de información estén habilitados para modificaciones sólo por las partes autorizadas. Las modificaciones incluyen escribir, modificar, cambiar el status, borrar, crear y demorar o responder mensajes enviados.
- *No Repudiabilidad* : Requiere que ni el emisor, ni el receptor de un mensaje este habilitado para denegar la transmisión.
- *Control de Acceso* : Requiere que el acceso a los recursos de la información sea controlado por o para el sistema destino (objetivo).
- *Disponibilidad* : Requiere que los componentes de un sistema de computación estén disponibles para autorizar a las partes cuando sea necesario.

Los diferentes tipos de ***ataques*** sobre la seguridad en un sistema de cómputos o una red se pueden caracterizar teniendo en cuenta la función de un sistema de cómputo como proveedor de información. En general, hay un flujo de información desde una fuente (origen), tal como un archivo o una región de la memoria principal, hacia un destino, tal como un archivo o un usuario. Desde este enfoque, podemos encontrar cuatro categorías generales de ataque:

- *Interrupción* (Ataque activo): un componente de un sistema es destruido o se vuelve inaccesible o no utilizable. Este es un ataque a la *disponibilidad.*
  Ejemplo: destrucción de una pieza del hardware, tal como un disco rígido, corte de la línea de comunicación o la deshabilitación del administrador de archivos.
- *Intercepción* (Ataque pasivo): una parte no autorizada obtiene acceso a un componente. Este es un ataque a la *confidencialidad.* Las partes no autorizadas pueden ser una persona, un programa o una computadora.
  Ejemplo: la copia ilegal de archivos y programas.
- *Modificación* (Ataque activo): una parte no autorizada no solo obtiene acceso sino que corrompe un componente. Este es un ataque a la *integridad.*

Ejemplo: cambiar valores a un archivo de datos, alterar un programa de manera que funcione diferente y modificar el contenido de los mensajes que son transferidos en una red.

- *Falsificación* (Ataque activo): una parte no autorizada inserta objetos falsos en el sistema. Este es un ataque a la *autenticidad.*
  Ejemplo: la inserción de mensajes ilegítimos en una red o la adición de registros en un archivo.

Los ***ataques pasivos*** responden a la naturaleza de los que escuchan o monitorean las transmisiones. El objetivo del oponente es obtener información que está siendo transmitida. Entre este tipo de ataques se encuentran la liberación de contenidos de mensajes y análisis del tráfico.

Los ***ataques activos*** involucran alguna modificación de la cadena de datos o la creación de una cadena falsa y puede ser subdividido en cuatro categorías: reproducción, enmascaramiento, modificación de mensajes y denegamiento de servicio.

No existe un **mecanismo** simple que provea todos los servicios arriba mencionados, sin embargo podemos notar que existe un elemento particular que es la base fundamental de la mayoría de los mecanismos en uso: las técnicas criptográficas. La encriptación y las transformaciones de información del tipo criptográficas son los mecanismos más comunes para proveer seguridad.

I. 1 - *Definiciones - Criptografía*

La ***criptografía*** es el arte y ciencia de hacer comunicaciones ininteligibles para todos, excepto para el apropiado receptor.

La palabra Criptografía se compone etimológicamente por las siguientes palabras: Kriptos (oculto) y Graphos (escribir). La criptografía estudia los procesos de encriptado y desencriptado de mensajes.

El ***criptoanálisis*** es el análisis de los mensajes encriptados o *criptogramas* con el objeto de descubrir la clave o el mensaje original.

Por lo tanto, la criptografía está relacionada con los métodos para preparar los criptogramas, y el criptoanálisis relacionado con las técnicas de ataque para violar el secreto de la información transmitida o almacenada. Al conjunto criptografía más criptoanálisis se lo denomina ***criptología***.

La criptografía es una competencia entre dos adversarios:

1. El *diseñador del sistema* especificando la familia de transformaciones
2. El *oponente* intentando recuperar el mensaje original y/o la clave y de este modo vencer el efecto del encriptado

Los métodos criptográficos han estado siempre fuertemente influenciados por la tecnología actual. La criptografía ha sobrevivido adaptándose a los cambios tecnológicos.

El ***encriptado*** es la transformación de datos en signos ilegibles para quien no disponga de la clave secreta para desencriptarlos. Su propósito consiste en asegurarle al usuario privacidad, ocultando la información a aquellos a quienes no está dirigida, e incluso a aquellos que tienen acceso a la información encriptada.

En una implementación multi-usuario el encriptado otorga seguridad a las comunicaciones que se producen a través de un medio inseguro.

La comunicación y la encriptación de información son procesos que tienen características comunes. Un canal de transmisión con ruido, por ejemplo la radio, el teléfono, etc. , sin

intencionalidad introduce distorsión durante la transmisión, por lo cual el mensaje recibido y el transmitido son generalmente diferentes.

En el proceso de encriptación el mensaje transmitido y el mensaje recibido son llamados ***plaintext*** y ***cyphertext*** respectivamente. El cyphertext es el resultado de una distorsión intencional del plaintext por una transformación criptográfica introducida con el fin de ocultar la información del mensaje original.

I. 2 - *Aplicaciones de la Criptografía*

Las dos aplicaciones fundamentales de la Criptografía son la protección de la información durante la transmisión y la protección de la información en su dispositivo de almacenamiento. En este último caso, la información es almacenada en forma encriptada previniéndose contra el robo o acceso al dispositivo.

En un entorno de tiempo compartido o de tipo transaccional multiusuario, la necesidad de protección entre los pares de puntos que sean potenciales comunicantes es elemental. Son de gran utilidad los esquemas de protección de clave pública con los protocolos de inicio de conexión y palabra clave de acceso al sistema.

También en el entorno multiusuario y especialmente en aquellas aplicaciones que envuelven una problemática financiera o bancaria o de transacciones comerciales, los esquemas de firma digital son imprescindibles para garantizar la autenticidad de las operaciones que se solicitan.

Actualmente, en pleno desarrollo de las aplicaciones en base a la comunicación a través del correo electrónico, se considera como necesaria la posibilidad de certificación de autenticidad y firma digital, para evitar violaciones de identidad y operaciones fraudulentas.

En general, es más importante la criptografía como herramienta de protección en las redes de teleproceso que en los sistemas de tiempo compartido, toda vez que en éstos últimos los datos más importantes están centralizados y pueden ser protegidos físicamente. En cambio, en una red de teleproceso, distribuida geográficamente, los nodos juegan un papel fundamental, ya que la información debe ser transmitida a través de los enlaces de comunicación, y en este caso la protección física no es posible.

I. 3 - *Historia de la Criptografía*

Como arte, la Criptografía data de los primeros tiempos de la historia registrada. Como ciencia se encuentra en una etapa preliminar de búsqueda de los criterios más adecuados de seguridad y medidas de complejidad de los métodos empleados.

El primer empleo de la escritura secreta o encriptada que se tiene constancia histórica es el caso de un esclavo que sirve como portador de un mensaje en la guerra entre Atenas y Esparta (siglo V a.c.). El general Lysandro recibe en una lista una serie de letras con aparente falta de sentido. Esta lista al ser enrollada en un determinado rodillo de madera mostraba longitudinalmente los símbolos dispuestos de tal manera que podía leerse el mensaje. El encriptado consistía en una simple alteración incluyendo símbolos innecesarios que desaparecen al desencriptar, enrollando la lista en el rodillo.

Otra de las primeras noticias que se tiene de la utilización de un método para comunicar información con carácter secreto data de la época de los romanos; la comunicación se efectuaba secretamente sustituyendo unos símbolos por otros en el conjunto de los que componían el mensaje, obedeciendo a una cierta regla fija. A este método se lo denomina método César o Caesar (siglo I a.c.).

César escribía a sus cónsules, generales y a Cicerón mediante un método que consistía en sustituir cada letra del mensaje por otra letra seleccionada de una forma fija.

Sin ninguna duda, la razón de la utilización del método César para la comunicación secreta, es de índole militar y diplomático, no existiendo otras razones adicionales hasta mucho tiempo después en la Edad Contemporánea.

Hasta el siglo XIV y época del Renacimiento hay un vacío en el progreso de la escritura secreta, como ocurrió con otros campos de las ciencias y las artes, resurgiendo con Cicco Simonetta, consejero y secretario de los duques Sforza en Milán entre 1375 y 1383. Su obra Liber Zifrorum es considerada como la primera y más antigua obra sobre criptografía que se conoce. En su obra estudia y analiza diversos sistemas fundamentados en sustituciones simples de letras y representaciones múltiples con adición de símbolos convencionales.

Un empuje decisivo fue dado a la criptografía por León Batista Alberti, contemporáneo del siglo XV, nacido en Génova y educado en Bolonia. Se le considera el padre de la criptología y fue un gran impulsor de los análisis criptoanalíticos.

El encriptado de Alberti consiste en un método sustitución simple de símbolos que se realiza utilizando dos círculos concéntricos con eje común, divididos en 24 casillas. En el anillo mayor se encuentran 20 letras del alfabeto (ordenadas alfabéticamente) más cuatro números (no aparecen H,J,K,U,Ñ,Y). En el anillo menor hay 23 letras, las 20 del exterior más la H,K,Y y la conjunción francesa et, en cualquier orden. El encriptado se efectúa enfrentando una casilla del anillo exterior con otra cualquiera del anillo interior. Esta pareja constituye la clave del encriptado, y con esta correspondencia entre ambos anillos se efectúa la encriptación. Bajo este sistema de encriptación resulta conveniente cambiar la clave cada cierto tiempo.

Durante el siglo XVI, el historiador y religioso benedictino alemán Trithemius, publica su obra "Poligraphiae" en la que además de afirmar que Carlomagno encriptaba sus comunicaciones, aportó dos métodos que estaban basados, uno en un sistema de varios alfabetos, denominado por su autor "Tabula recta" y otro en una sustitución de letras por palabras, al revés que el método de Lavinde.

Dentro del mismo período renacentista italiano, otros estudiosos de la criptografía como Giovanni Belasso y Girolamo Cardano nacido en Pavía, quien además de sus estudios en el campo de las matemáticas y de la física, aportó en el campo de la criptografía un sistema basado en una carta o tarjeta con agujeros perforados, de tal manera que al colocarla sobre un determinado texto preconcebido se obtenía el mensaje original.

El fin del siglo XVI y todo el siglo XVII son épocas muy activas respecto al uso de la criptografía, pero sin progresos evidentes en la aparición de nuevos métodos. Lo que sí es importante es la investigación criptoanalítica en el hallazgo de claves y desencriptación de correspondencia. John Wallis fue el primer gran criptoanalista, siendo famosa su solución a la correspondencia entre el francés Luis XIV y su embajador en Polonia, en 1689.

El siglo XIX produce la consolidación de un método de encriptación que consiste en la alteración del orden de los símbolos del mensaje, que previamente se ha subdividido en bloques de longitud fija llamado transposición. Sir Charles Wheaststone autor de numerosos inventos, entre ellos el puente que lleva su nombre para medir resistencias eléctricas, propuso algunos sistemas basados en métodos de transposición.

Napoleón utilizó varios sistemas de encriptación, basados en el sistema denominado Richelieu y Rossignol. También utilizó el sistema ya clásico, de asignación de símbolos numéricos a grupos de una o más letras. Napoleón usó un sistema con no más de 200 grupos numéricos y no dio la clave más que a sus mariscales.

El método Rossignol se basaba en el empleo de 587 grupos silábicos con la utilización de símbolos especiales. El método de Richelieu está fundamentado en la utilización del método de transposición, incrementando cada bloque con alguna cantidad variable.

No obstante, el mayor desarrollo de la criptografía se ha realizado durante el siglo XX con la utilización comercial del telégrafo y los impulsos proporcionados en las dos guerras

Mundiales para buscar métodos de comunicación secretos, siendo de interés el desarrollo de sofisticadas máquinas de cifrado durante la Segunda Guerra Mundial.

El progreso del criptoanálisis fue formidable, poseyendo los Estados Unidos su propio departamento criptoanalítico. Durante la Primera Guerra Mundial, los franceses consiguieron averiguar la clave del cifrado alemán ADFGX y los ingleses el telegrama ZIMMERMAN [Way, P. 77], el cual usaba un código que asignaba cifras a las palabras de acuerdo con un libro de códigos que poseían el emisor y el receptor del mensaje. El ADFGX usaba únicamente esas cinco letras para sustituir una combinación de ellas por cada símbolo del mensaje original. El encriptado consistía en la sustitución de cada letra por un par de símbolos, de acuerdo con una matriz.

	A	**D**	**F**	**G**	**X**
A	n	b	x	r	u
D	q	o	k	d	v
F	a	h	s	g	f
G	m	z	c	l	t
X	e	i	p	j	w

El sistema se completa con una transposición de bloques con longitud 20. Sin embrago, los criptoanalistas emplearon la herramienta proporcionada por la frecuencia de las letras para vulnerar el sistema.

Posteriormente el sistema fue ampliado con un símbolo más, la V , convirtiéndose en el cifrado ADFGVX, que permitió encriptar los números. A partir de 1930, aparecen máquinas de cifrado que efectúan encriptaciones muy complejas, comparadas con las existentes hasta esa época. Además, la aparición de trabajos sobre criptoanálisis, como el de los hermanos Friedman, apoyó la necesidad de construir métodos de encriptación que puedan escapar a los conocimientos criptoanalíticos existentes hasta el momento.

Las máquinas que reciben el nombre de PURPLE y ENIGMA fueron las precursoras de las utilizadas en la Segunda Guerra Mundial, que fueron notablemente mejores que las usadas durante la primera mitad de los años 30. A partir de este momento, el incremento continuado

en el uso de las telecomunicaciones y la comercialización de computadoras desde los 50, amplían el ámbito de la criptografía mejorando las técnicas de encriptación y proporcionando potentes herramientas en los análisis criptoanalíticos.

A partir de1949, Claude E. Shannon [Shannon, 49] aporta los principios matemáticos referentes a esta materia, siendo la base teórica aplicada al desarrollo de las comunicaciones.

A comienzo de los años 50 se produce un cambio fundamental en la práctica de la criptografía. La comercialización de las computadoras y la potencia de cálculo que aportaron éstas, promovió la aparición de métodos de encriptación que se basan en la dificultad computacional para quebrantarlos. La velocidad de cálculo proporcionó una solución más rápida y flexible para efectuar el proceso de encriptación y desencriptación, pero al mismo tiempo aumentó las posibilidades de vulnerar la información.

Por último, desde la década de los sesenta, con la utilización masiva de las computadoras y la gran potencia de cálculo aportada, se empiezan a utilizar métodos criptográficos basados en técnicas que se denominan computacionales, cuyo fin es hacer altamente improbable la vulnerabilidad del sistema mediante el empleo de métodos con una elevada complejidad computacional, haciendo necesario incluso con las técnicas actuales, mucho tiempo de cálculo para hacer frágil e inseguro al sistema criptográfico. El esfuerzo reside concretamente en la complejidad computacional para resolver ciertos problemas matemáticos con los métodos actualmente conocidos.

I. 4 - *Estructura de un Sistema Secreto*

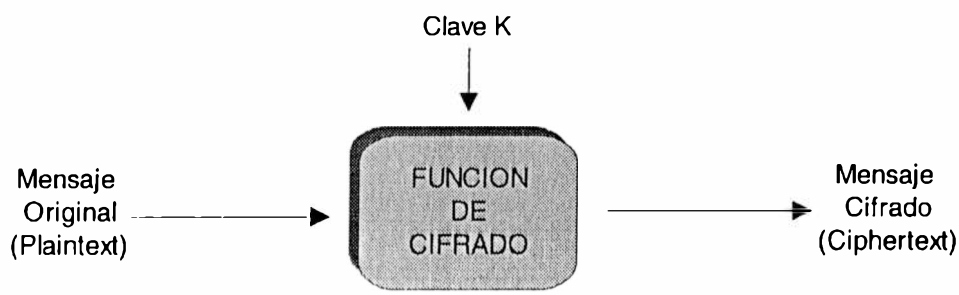
La protección de la información en un sistema automático de tratamiento de la información se aplica mediante técnicas criptográficas, tanto a los datos almacenados en

soporte, discos o cintas, como a la información transmitida a través de un canal de comunicación.

De esta forma, un acceso indebido a los soportes, o una escucha realizada interceptando un canal de comunicación presentará información en forma encriptada, resultando incomprensible al que no posea el resto de la información necesaria para su desencriptación.

Es preciso destacar que la aplicación de técnicas criptográficas al almacenamiento de la información no previene contra la destrucción física, pero disminuye fuertemente el riesgo de su obtención no controlada.

En un sistema secreto hay dos funciones elementales que intervienen, que se denominan función de cifrado o proceso de encriptación y función de descifrado o proceso de desencriptación. Estas funciones se suponen conocidas por el oponente. Si el sistema fuera dependiente únicamente de las funciones de cifrado y descifrado, el oponente tendría suficiente información para vulnerar el sistema, por tanto es preciso la existencia de algo que siendo desconocido impida al oponente obtener la información. Esto es la clave, necesaria para descifrar el mensaje.



La función de encriptación *E*, combina de alguna manera el mensaje original con la clave *K*, para obtener el mensaje encriptado. Consecuentemente, la función de desencriptación *D*, combina el mensaje encriptado con la clave *K*, para obtener el mensaje original.

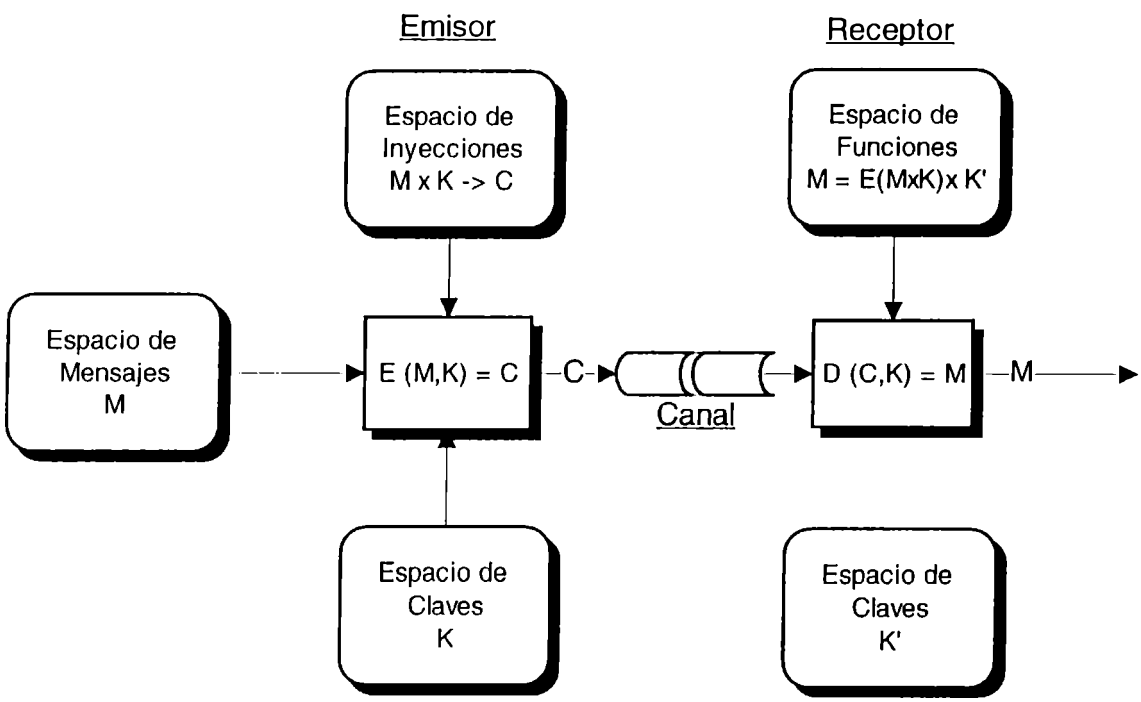
Por lo tanto, siendo *M* el mensaje original y *K* la clave, el mensaje encriptado *C*, será :

$$C=E(M,K)$$

que expresado en forma de aplicación es :

$$(M \times K) \rightarrow C$$

donde se ha designado a M como el espacio de mensajes originales, a K como el espacio de claves, y $M \times K$ el espacio de todas las inyecciones $M \times K$.



Análogamente, designando a D como función de desencriptación :

$$M = D(C,K) = D(E(M,K),K)$$

que expresado en forma de aplicación es :

$$(C \times K) \rightarrow M$$

o lo que es igual :

$$((M \times K) \times K) \rightarrow M$$

La criptografía tradicional se basa en el concepto de que tanto el que envía el mensaje como el que lo recibe conocen y utilizan la misma clave secreta. Por lo tanto el emisor utiliza una clave secreta para encriptarlo y el receptor del mensaje utiliza la misma clave secreta para desencriptarlo. A este tipo de esquema se lo conoce como sistema criptográfico **simétrico.**

Por otro lado, los esquemas criptográficos **asimétricos** utilizan dos claves, una pública y otra privada, que son aplicadas por el emisor y el receptor para la encriptación y desencriptación de mensajes. Esto quiere decir que una clave es utilizada para encriptar el mensaje y la otra para desencriptarlo.

Dado que toda la seguridad reside en mantener el secreto de la clave, se debe poseer un método seguro de distribución de claves.

Referencias

- [Shannon, 49] Shannon, C. "Comunication Theory of Secrecy Systems". Bell Syst. Tech. J. Vol. 28. 1949.

- [Way, P. 77] Way, Peter "Codes and Ciphers". Aldus Books Limited. 1977. U.K.

II - METODOS CRIPTOGRAFICOS CLASICOS

La aplicación de métodos criptográficos es la solución universalmente aceptada para evitar los actos que puedan vulnerar la información y evitar cualquier utilización no autorizada de la información.

II. A. METODOS POR SUSTITUCION

En estos métodos cada símbolo del alfabeto del texto original es reemplazado por otro símbolo de ese alfabeto.

II. A - 1 SUSTITUCION MONOALFABETICA

Un método es de sustitución monoalfabética cuando la sustitución sólo depende del símbolo del alfabeto.

METODO CAESAR

Es un método utilizado por los romanos y atribuido a Julio César. Originalmente la encriptación por el método César consistía en una clave simple con la siguiente transformación única:

$$C: M \rightarrow M + 3$$

Posteriormente fue generalizado a un sistema de encriptación con 26 claves, $0 < K \leq 26$, correspondientes a los 26 desplazamientos cíclicos del alfabeto:

$$C_k: M + K$$

El número de claves, al ser tan pequeño, facilita la labor del criptoanalista, haciendo vulnerable el sistema. La generalización del método de César da lugar al método de encriptación por simple sustitución o sustitución monoalfabética.

El método César sustituye cada letra del mensaje original, por otra letra que formará parte del criptograma, de acuerdo con el siguiente procedimiento:

1. A cada letra del alfabeto correspondiente al lenguaje natural se le asigna un número secuencialmente.
2. La correspondiente letra del mensaje encriptado se obtiene desplazándose tres posiciones con respecto al número que tiene la letra del mensaje original.

La generalización posterior con la posibilidad de utilizar 27 claves en lugar de una clave única, aplica un desplazamiento para obtener la correspondiente letra del criptograma que puede ser de 0 a 26. El procedimiento de sustitución es el mismo que el del método César, a excepción de que en el segundo punto, en vez de realizar un desplazamiento de 3 posiciones, se desplaza K posiciones.

Criptoanálisis

El estudio de este sistema por un criptoanalista, probablemente terminaría con éxito seguro, ya que sólo existen 26 correspondencias posibles y por lo tanto resulta relativamente fácil realizar un estudio exhaustivo de las claves para encontrar la utilizada. Por otro lado, teniendo en cuenta los estudios efectuados sobre el comportamiento de las letras en los lenguajes naturales que indican una probabilidad estable en la aparición de cada letra, el criptoanalista sólo tendría que esperar capturar un número suficiente de mensajes para establecer una frecuencia de las letras del criptograma, y por semejanza en el comportamiento del lenguaje natural hallar la clave.

SUSTITUCION SIMPLE

Permite que cualquier permutación del alfabeto sea utilizada como clave de sustitución en base a un sistema letra por letra.

El sistema de sustitución simple, también conocido como sustitución monoalfabética, aplica una transformación que es una congruencia lineal, cuya representación general es del tipo:

$$C: M \rightarrow YM + K$$

donde M es el valor numérico asignado a cada letra del alfabeto correspondiente al mensaje original, Y es una constante para determinar una correspondencia concreta entre todas las posibles para el conjunto de letras del alfabeto del criptograma y el alfabeto del mensaje original. K es una constante para fijar un desplazamiento entre las letras de uno y de otro alfabeto dentro de la correspondencia determinada.

Ejemplo:

Dado el mensaje FECHA CAIDA DIA D, utilizando un desplazamiento K=7, el mensaje encriptado sería: LEQYD QDFXD XFD X.

Criptoanálisis

El número de claves es $27! = 10^{28}$, que es suficientemente alto como para que el método de análisis exhaustivo de todas las claves posibles no pueda ser seguido fácilmente por un criptoanalista, aunque es posible realizarlo mediante un análisis estadístico. Para contrarrestarlo debería cambiarse la clave frecuentemente.

Asimismo, si por algún sistema se consigue obtener la equivalencia entre una letra del alfabeto de los mensajes originales y su correspondiente del mensaje encriptado o criptograma, quedan encontrados los parámetros Y y K, haciendo plenamente vulnerable el sistema. Esto dio origen en su momento a la búsqueda de otros sistemas más seguros para encriptar la información.

ENCRIPTACION DE HOMOFONICOS

Es un método de encriptación por sustitución, de tal modo que cada símbolo en el alfabeto del mensaje original es correspondido por uno de un conjunto de símbolos denominados homofónicos.

Ejemplo:

A cada símbolo del alfabeto español se le asigna al azar un cierto número de enteros entre 0 y 99, de tal modo que el número de enteros asignados es proporcional a la frecuencia relativa de cada letra, y ningún entero es asignado a más de una letra:

A: 17,19,34,41,56,60,67,83
I: 08,22,53,65,88,90
L: 03,44,76
N: 02,09,15,27,32,40,59
O: 01,11,23,28,42,54,70,80
P: 33,91
T: 05,18

Mensaje Original: PILOTO

Mensaje Encriptado: 33, 08, 03, 01, 05, 11

II. A - 2 SUSTITUCION POLIALFABETICA

Un método es de sustitución polialfabética cuando la sustitución depende además del símbolo del alfabeto, de su posición.

METODO VIGENERE

Una solución a los problemas de vulnerabilidad que plantean los sistemas monoalfabéticos es usar no sólo un alfabeto sino varios, utilizando una clave que especifique qué sustitución va a realizarse para cada símbolo en el proceso de encriptación.

El sistema más simple de sustitución polialfabética es el método de Vigenere, donde la clave es una palabra o frase que se repite las veces necesarias para encriptar el mensaje.

Ejemplo:

Sea la clave CLAVE y el mensaje LOS CIELOS SE CIERRAN SOLOS, el esquema de encriptación utilizando el método Vigenere sería el siguiente:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z	(1)
C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	(2)
L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	(3)
A	B	C	D	E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z	(4)
V	W	X	Y	Z	A	B	C	D	E	F	G	H	Y	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	(5)
E	F	G	H	I	J	K	L	M	N	Ñ	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	(6)

Palabra Clave = CLAVE

L	O	S	C	I	E	L	O	S	S	E	C	I	E	R	R	A	N	S	O	L	O	S	(7)
C	L	A	V	E	C	L	A	V	E	C	L	A	V	E	C	L	A	V	E	C	L	A	(8)
N	Z	S	X	M	G	V	O	Ñ	W	G	N	I	Z	V	T	L	N	Ñ	S	N	Z	S	(9)

Referencias:

(1) Alfabeto del mensaje original
(2) Primer alfabeto, correspondiente a la letra 1 de la clave
(3) Segundo alfabeto, correspondiente a la letra 2 de la clave
(4) Tercer alfabeto, correspondiente a la letra 3 de la clave
(5) Cuarto alfabeto, correspondiente a la letra 4 de la clave
(6) Quinto alfabeto, correspondiente a la letra 5 de la clave
(7) Mensaje original o Plaintext
(8) Clave repetida
(9) Mensaje encriptado o Cyphertext

Puede observarse en el criptograma (línea 9) la vulnerabilidad de este sistema, dado que aparece repetida la secuencia NZS en el mensaje encriptado, correspondiente a ubicaciones iguales de letras del mensaje original y la clave. Por lo tanto, el criptoanalista que trata de interceptar el mensaje, podría encontrar parejas de mensaje y clave que se repiten, produciendo los mismos símbolos en el mensaje encriptado, tal como surge del ejemplo precedente.

Para evitar la exposición a estas vulnerabilidades utilizando este método, podría usarse una clave suficientemente larga como para que no se repita al ubicarlo debajo del mensaje original, y de esta forma evitar la repetición de símbolos en el mensaje encriptado. Sin embargo, el criptoanalista tiene otras salidas para descubrir el criptograma, incluso aunque no se repita la clave.

Criptoanálisis

El uso de este método permite al criptoanalista analizar los mensajes basándose en dos aspectos:

1. La suposición de que tanto la clave como el mensaje original tienen idéntica distribución en los símbolos, dado que son frases cuyas palabras pertenecen a un determinado lenguaje natural.
2. Un método denominado "palabra probable", según el cual el criptoanalista extrae del mensaje encriptado o criptograma, palabras que considera más probables dentro del texto, realizando esto sucesivamente hasta que vaya configurando la clave.

METODO VERNAM

Una variante del sistema de sustitución polialfabética fue ofrecida por el ingeniero americano G. S. Vernam en 1918, diseñado inicialmente para transmisión telegráfica, empleando una palabra clave aleatoria de longitud infinita.

Vernam reconoció que si una secuencia aleatoria se añadía módulo 2 al mensaje, entonces las características de frecuencia o probabilidad de los símbolos, que habían constituido las herramientas fundamentales empleadas por los criptoanalistas para vulnerar los sistemas de Vigenere, no podían ser utilizadas. La aproximación casi intuitiva de Vernam fue probada veinte años después por Shannon.

II. B. METODOS POR TRANSPOSICION

La encriptación de mensajes por transposición se efectúa operando en bloques de longitud n, empleando como claves las $n!$ permutaciones posibles transponiendo los símbolos del mensaje.

Para encriptar utilizando este método se deben seguir los siguientes pasos:

1. Se divide el mensaje original en bloques de longitud n, resultando un cierto número de grupos de n símbolos de longitud.
2. Se aplica a cada grupo de longitud n, la transposición determinada por la clave particularmente seleccionada.

Generalmente los espacios separados de palabras son suprimidos para evitar dar facilidades al criptoanalista.

Ejemplo:

Sea n=6 la longitud de los grupos y la clave seleccionada, una de las 6! permutaciones posibles, en este caso la clave será 651342.

La letra o símbolo que ocupe el lugar 6 de cada grupo del mensaje original deberá ser puesta en la posición 1 del correspondiente grupo del criptograma; la que ocupe el lugar 5 pasará a la posición 2 y así sucesivamente.

Posición Original:	1	2	3	4	5	6
Mensaje Original:	P	L	A	N	O	S
Mensaje Encriptado:	S	O	P	N	A	L

1	2	3
D	E	L
H	O	D

4	5	6	1	2	3
C	O	H	E	T	E
L	C	E	Z	Y	E

4	5	6
X	Y	Z
E	X	T

Criptoanálisis

Desde el punto de vista del criptoanálisis, este sistema continúa manteniendo la frecuencia de las letras simples del lenguaje, permitiendo su estudio por métodos estadísticos, pero destruye las estadísticas de otro orden como diagramas o bigramas, trigramas, etc. Esto proporciona mejores resultados que el encriptado por simple sustitución cuando se trata de lenguajes naturales.

III. - CRIPTOGRAFIA CON CLAVE SECRETA

La Criptografía tradicional se basa en el concepto de que tanto el que envía el mensaje como el que lo recibe conocen y utilizan la misma *clave secreta.* El que envía el mensaje utiliza una clave secreta para encriptarlo y el que lo recibe utiliza la misma clave para desencriptarlo. Este método se conoce como criptografía con clave secreta.

El principal problema de este método, consiste en conseguir que ambas partes conozcan la misma clave sin que ningún tercero se entere. Si se encuentran geográficamente alejados, deberán confiar en un correo privado, sistema telefónico o cualquier otro método de transmisión para que la clave se mantenga en secreto. Si la clave es interceptada, quien la conozca podrá luego utilizarla para leer todos los mensajes encriptados.

La producción, transmisión y almacenamiento de las claves se denomina administración de claves, todos los sistemas criptográficos deben lidiar con la seguridad de esta administración.

La criptografía con clave secreta ha tenido a menudo dificultades para brindar la seguridad necesaria en este aspecto.

DATA ENCRYPTION STANDARD (DES)

III. 1 - Introducción

El 15 de enero de 1977 la National Bureau of Standards (NBS) del Departamento de Comercio de USA publicó su Federal Information Processing Standards Publication con el título de Data Encription Standard, el que fue aceptado como estándar y todas las computadoras vendidas al gobierno de los EEUU están obligados a incluir la capacidad de utilizar este estándar. Este procedimiento es la conclusión del estudio y desarrollo de un método criptográfico realizado durante los años 1968 y 1975 por una importante empresa fabricante de computadoras. El DES consiste básicamente en 16 pasos o ciclos de sustitución y permutación de bits, controlados por una clave.

III. 2 - Descripción del Algoritmo DES

DES es un sistema criptográfico simétrico de clave secreta. Cuando se utiliza para comunicaciones, tanto el remitente como el receptor deben conocer la clave secreta que se usa tanto para encriptado como para el desencriptado del mensaje. El DES puede ser utilizado también por un solo usuario para guardar archivos encriptados en un disco rígido. Si se utiliza en un medio de usuarios múltiples, distribuir la clave con seguridad puede llegar a ser un serio inconveniente.

DES fue diseñado para ser implementado en hardware y opera con relativa rapidez, permitiendo encriptar grandes cantidades de información.

El algoritmo DES transforma un bloque de texto original de 64 bits en un bloque de texto encriptado bajo el control de una clave de 64 bits de los cuales 56 bits son usados directamente por el algoritmo y 8 son utilizados para detección de errores, como bits de

paridad impar de cada byte, es decir si existe un número impar de bits 1 en cada byte. Si se utiliza la misma clave el procedimiento de desencriptado convierte los datos a su forma original.

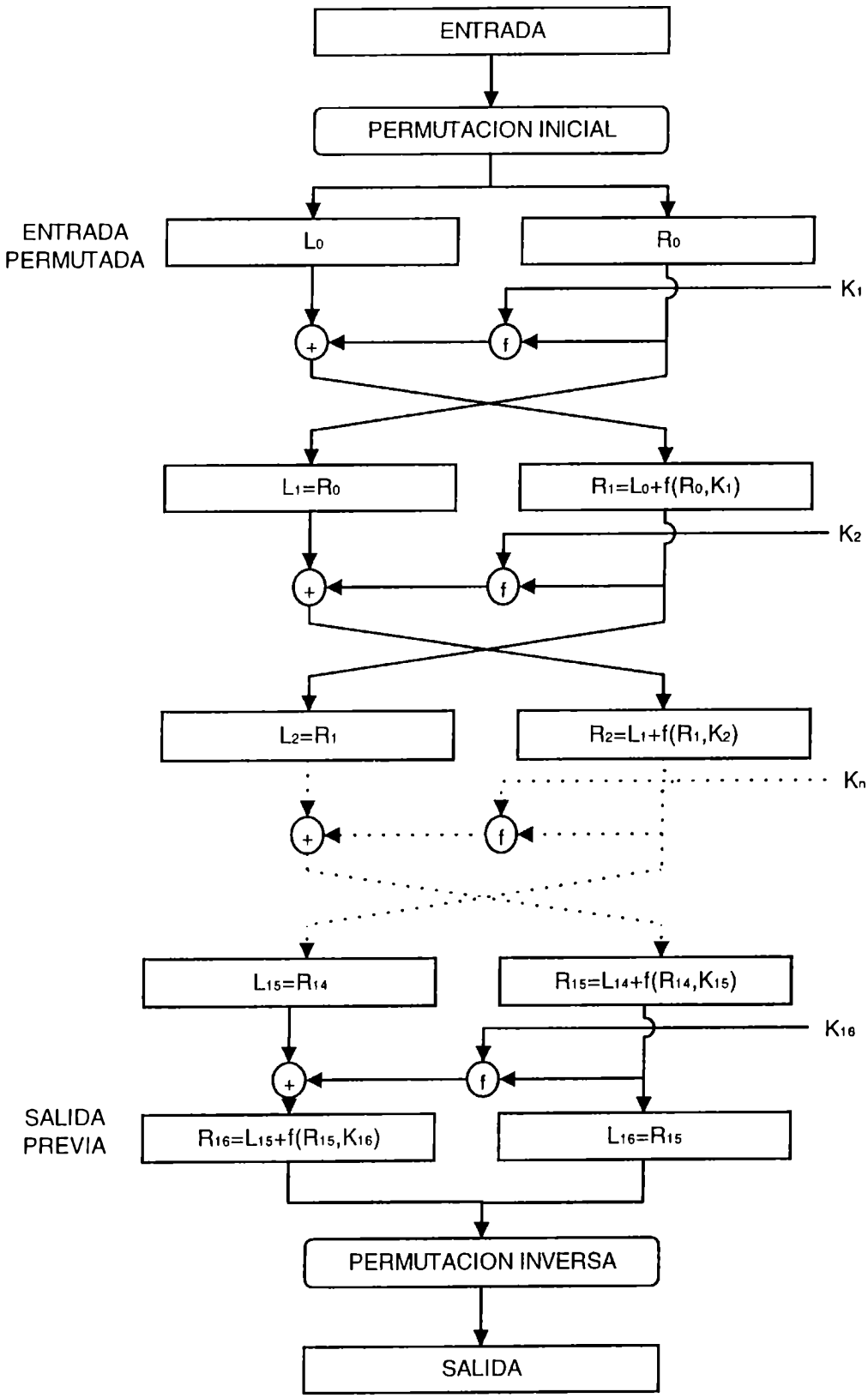
Todos los usuarios conocedores de la clave están en condiciones de encriptar o desencriptar mensajes con el resto de los miembros del grupo que posean la clave y el algoritmo. Consecuentemente y toda vez que los algoritmos de encriptación son públicamente conocidos, la seguridad de la información depende solamente de la seguridad con que se mantenga en secreto la clave utilizada.

El bloque a encriptar se somete a una permutación inicial de sus bits denominada *IP* y a continuación se inicia un cálculo complejo cuyo resultado depende no sólo de la entrada sino de la clave, de tal modo que existe una interacción entre clave y datos. Después de este cálculo que son 16 operaciones distintas, se efectúa una nueva permutación que es la inversa de la inicial denominada IP^{-1}.

El cálculo de 16 operaciones puede definirse simplemente en términos de una función f denominada función de encriptación y una función *KS* denominada planificación de clave.

Por lo tanto, al bloque de entrada de 64 bits, que llamaremos T, se le aplica una permutación inicial *IP* obteniéndose $T_0 = IP(T)$.

Después de pasar por 16 iteraciones de la función f, se aplica la permutación inversa IP^{-1} para obtener el resultado final.


ENTRADA
PERMUTACION INICIAL
ENTRADA PERMUTADA
L0
R0
K1
+
f
L1=R0
R1=L0+f(R0,K1)
K2
L2=R1
R2=L1+f(R1,K2)
Kn
L15=R14
R15=L14+f(R14,K15)
K16
SALIDA PREVIA
R16=L15+f(R15,K16)
L16=R15
PERMUTACION INVERSA
SALIDA

donde Ti es el resultado de la i-ésima iteración, L_i y R_i es la mitad izquierda y mitad derecha del bloque de 64 bits en los que se divide T_i ($T_i = L_i\ R_i$), siendo i el número de ciclo (0 a 16). Por lo tanto cada L_i y R_i se define como sigue:

$$L_i = t_1 \ldots\ldots t_{32}$$

$$R_i = t_{33} \ldots\ldots t_{64}$$

Entonces:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

donde $\oplus$ es un Or-Exclusivo o la operación binaria suma módulo dos y K_i es una clave de 48 bits elegidos de la clave secreta de 64 bits.

Después de las iteraciones efectuadas, el bloque de presalida es:

$$R_{16}L_{16}$$

La desencriptación se realiza aplicando el mismo algoritmo y clave usados para encriptar el bloque de entrada. Asimismo, en cada iteración hay que tener cuidado en utilizar el mismo bloque de bits de la clave secreta que fuera utilizado durante el proceso de encriptación.

Utilizando la misma notación, resultará:

$$R_{i-1} = L_i$$

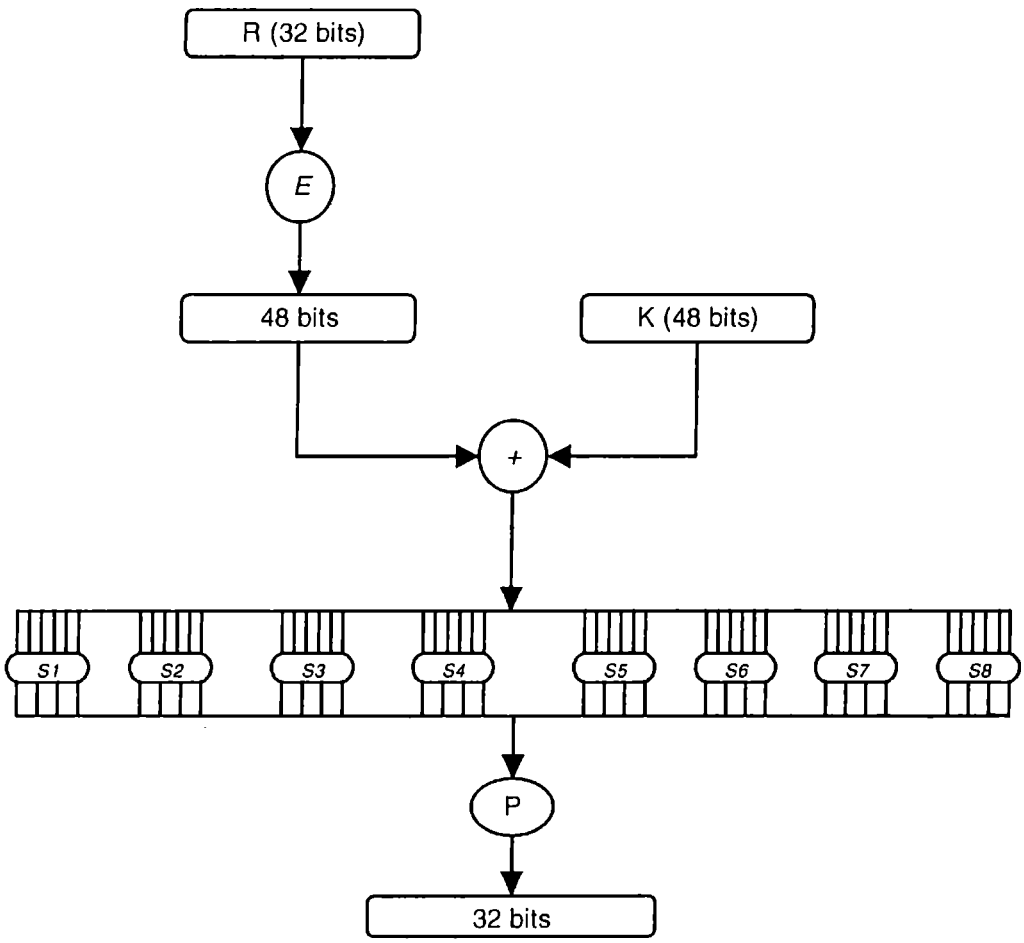
$$L_{i-1} = R_i\, f(L_i, K_i)$$

Ahora $R_{16}L_{16}$ es el bloque de entrada permutado con el que se inicia el proceso de las 16 iteraciones y L_0R_0 es el bloque de presalida.

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

III. 3 - Descripción de la Función de Encriptación

La función de encriptación f se explica en términos de una función de expansión E, la operación suma módulo dos, ocho funciones de selección S_i y una permutación P, tal como se ilustra en la figura.



La función de expansión E toma un bloque de 32 bits y obtiene un bloque de 48 bits como salida. Este proceso es una permutación de bits de tal modo que numerando los bits del bloque de entrada de izquierda a derecha 1,2,...,31,32, el bloque de salida contiene los bits del bloque de entrada, pero en el siguiente orden:

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Orden de los bits permutados

Una vez obtenido un bloque de 48 bits resultante de la función E, se efectúa la operación suma módulo dos con K_i, obteniéndose un nuevo bloque de 48 bits, que se subdivide en grupos de 6, teniendo por lo tanto ocho bloques de 6 bits que van a ser los argumentos de las funciones S_i que denominaremos cajas.

Las funciones S_i toman un bloque de 6 bits y producen un bloque de 4 bits. Las ocho funciones S_i producen entonces 32 bits que serán objeto de una permutación P.

A continuación y con el objeto de ilustrar como funcionan las cajas S_i describiremos la caja S_1, siendo las siete restantes de la misma estructura aunque con distintos valores en las tablas que usan.

Sea A el bloque de bits de entrada a S_i. Sean a_1, a_2, a_3, a_4, a_5, a_6 los seis bits de A, se obtiene $S_1(A)$ tomando los bits a_1 y a_6 de modo tal que a_1 a_6 representan un número en binario cuyo rango es 0 a 3. Sea ese número i.

Tomando a continuación los bits a_2 a_3 a_4 a_5, su correspondiente número en binario está en el rango 0 a 15. Sea ese número j.

Existen ocho tablas predefinidas, una distinta para cada función S_i con valores dispuestos en 4 filas numeradas de 0 a 3 y 16 columnas de 0 a 15.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Tabla para S_1

Los valores *i* y *j* obtenidos precedentemente son usados como coordenadas en la fila y columna respectivamente, determinando un número en la tabla, cuyo valor máximo es 15, por tanto se representa en 4 bits, que es la salida de las funciones S_i.

Los 32 bits correspondientes a $S_1\ S_2\ S_3\ S_4\ S_5\ S_6\ S_7\ S_8$ son el bloque de partida para realizar la permutación *P* cuyos bits se ordenan del siguiente modo:

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Permutación P

Finalmente, la notación formal de la función *f* siguiendo la explicación anterior y siendo $b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7\ b_8$ los ocho bloques de 6 bits cada uno, es la siguiente:

$$b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7\ b_8 = K_i \oplus E(R_{i-1})$$

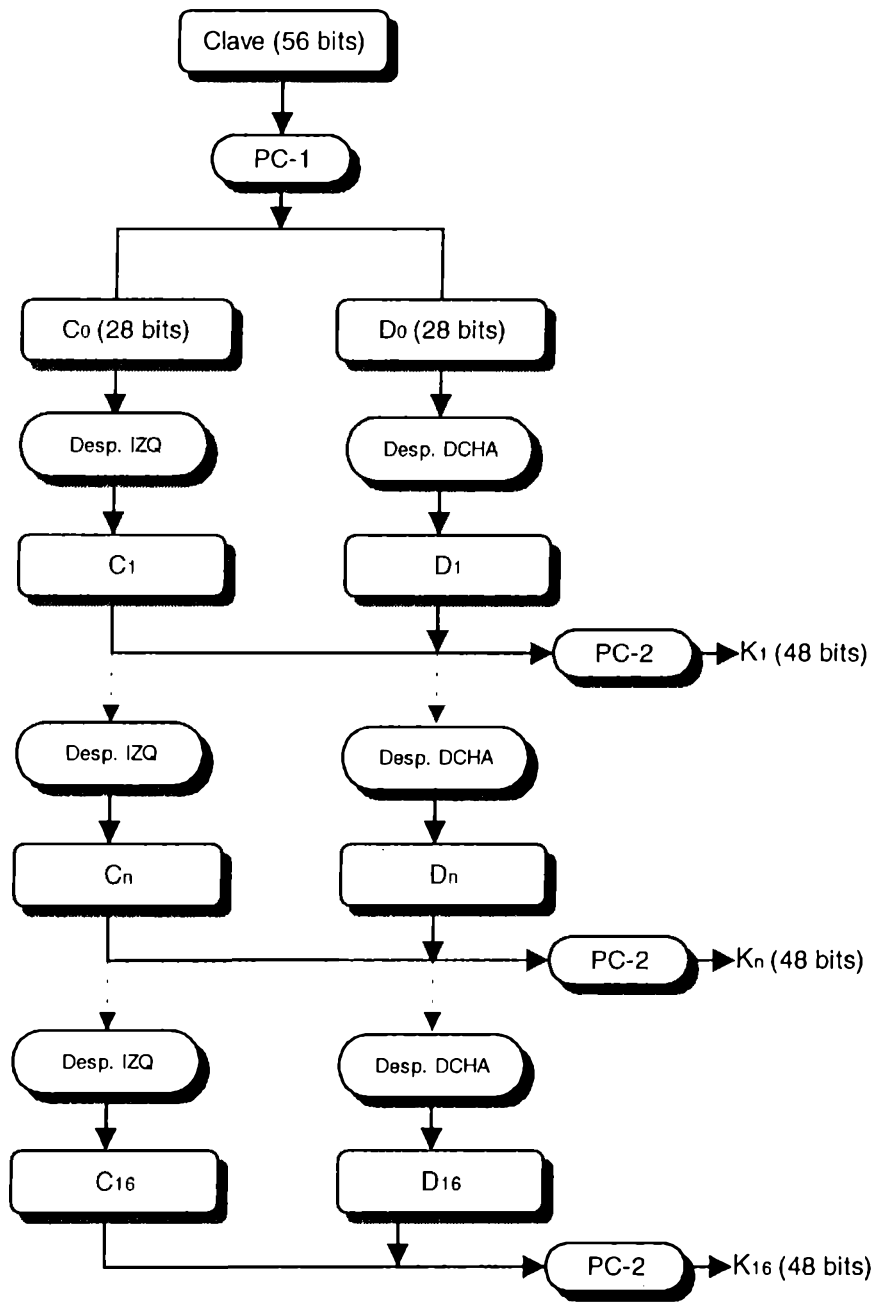
El bloque final $f(R_{i-1}, K_i)$ se define como sigue:

$$f(R_{i-1}, K_i) = P(S_1\ (b_1)\ S_2\ (b_2).....S_8\ (b_8))$$

siendo *P* la permutación final.

III. 4 - Descripción de la Función Planificación de clave (KS)

La función de planificación de clave proporciona durante los 16 pasos que tiene, las 16 K_i para ser usadas en el ciclo i-ésimo del cálculo de la función f(R,K). En la figura se observan los distintos pasos que la función *KS* debe realizar para proporcionar las dieciséis claves.



Primeramente la clave secreta sufre una permutación de sus bits, denominada Permited Choice-1 (PC-1). Esta permutación no usa los bits que son múltiplos de 8, como puede deducirse de la ordenación resultante:

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

PC-1

Una vez efectuada la permutación, los 56 bits se dividen en dos mitades de 28 bits cada una, que llamaremos C_i y D_i tomando i valores entre 0 y 16. C_0D_0 es el primer conjunto de bits obtenido directamente de la permutación PC-1. La obtención de los bloques subsiguientes se hace siempre partiendo del bloque anterior.

Así, el bloque C_1D_1 se obtiene del bloque C_0D_0 efectuando a cada mitad un proceso de rotaciones a la izquierda de acuerdo a la siguiente tabla:

Nro de Ciclo	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Nro de bits a rotar	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Finalmente, cada uno de los bloques C_iD_i, para valores de i entre 1 y 16, es sometido a una nueva permutación denominada PC-2 que proporciona para cada uno de los ciclos, la clave K_i a utilizar por la función *f*. Es de notar que la permutación PC-2 recibe como entrada 56 bits y se obtienen 48 bits de salida, que es la misma longitud de salida de la función expansión *E*, en el cálculo de $f(R,K)$.

III. 5 - Reversibilidad del Algoritmo de Encriptación

Sean C_1 a C_{16} y D_1 a D_{16} las iteraciones de los procesos de encriptación y desencriptación respectivamente. Las ecuaciones de la encriptación en la iteración 16, es decir C_{16} son:

$$R_{16} = L_{15} f(R_{15}, K_{16}) \quad (1)$$

$$L_{16} = R_{15} \quad (2)$$

luego sustituyendo (2) en (1)

$$R_{16} = L_{15} f(L_{16}, K_{16}) \quad (3)$$

donde al despejar L_{15}

$$L_{15} = R_{16} f(L_{16}, K_{16}) \quad (4)$$

que es el paso D_1 del proceso de desencriptación, ya que la obtención de

$$R_{15} = L_{16}$$

es inmediata.

Este comportamiento es idéntico para los 16 pasos, de tal modo que en el primer paso de desencriptación se obtiene el mismo bloque de información que en el último del proceso de encriptación, resultando equivalentes los siguientes pasos:

$$D_1 = C_{16}$$

$$D_2 = C_{15}$$

$$\vdots$$

$$D_{15} = C_2$$

$$D_{16} = C_1$$

Asimismo, las permutaciones IP y IP^{-1} son definidas como reversibles, por lo tanto los procesos de encriptación y desencriptación son totalmente inversos si se siguen los pasos tal como se expusieron.

III. 6 - Criptoanálisis

Nunca se ha "quebrado" el sistema DES a pesar de los esfuerzos de los investigadores durante los últimos años. El método obvio de ataque consiste en forzar una exhaustiva búsqueda del espacio de la clave lo cual toma 2^{55} operaciones de promedio. Tiempo atrás, se sugirió [Diffie and Hellman, 1977] que un oponente rico y poderoso podría construir una computadora especial para quebrar el sistema DES en un lapso razonable. Estas ideas sembraron dudas en cuanto a la seguridad del sistema DES.

A pesar de las sospechas, no existe un modo posible de quebrar el DES más rápidamente que por medio de la búsqueda exhaustiva. El costo de una computadora especial para realizar una búsqueda exhaustiva se calcula en un millón de dólares [Wiener, 1993].

Sin embargo, recientemente, Eli Biham y Adi Shamir anunciaron el primer ataque al DES que es mejor que la búsqueda exhaustiva, utilizando una nueva técnica que se conoce como criptoanálisis diferencial. Este ataque requiere un encriptado de 2^{47} textos planos elegidos cuidadosamente, es decir elegidos por el atacante. Aunque es solamente un avance teórico, este ataque no podría llevarse a cabo en la práctica bajo circunstancias normales porque requiere que el atacante tenga fácil acceso al dispositivo DES para encriptar los textos por él elegidos. Otro ataque, conocido como criptoanálisis lineal [Matsui, 1993] no requiere de estos textos elegidos.

El consenso indica que DES, utilizado de forma apropiada, es seguro contra todo enemigo, excepto los más poderosos. DES se utiliza en una gran variedad de sistemas criptográficos, y prácticamente todos los sistemas criptográficos con clave pública lo utilizan en algún nivel.

III. 7 - Consideraciones

Cuando se utiliza el sistema DES se deben cambiar las claves con frecuencia para prevenir ataques que requieran un sostenido análisis de la información. En un contexto de comunicaciones, se debe encontrar la forma de transmitir las claves con seguridad entre el remitente y el receptor. El uso del sistema RSA u otra técnica de administración de clave, resuelve ambos problemas: se genera una clave DES diferente para cada sesión y la administración de claves segura lo proporciona el encriptado de la clave DES con la clave pública RSA del receptor. Bajo estas circunstancias, el sistema RSA se puede usar como una herramienta para mejorar la seguridad del DES (o de cualquier otro cifrado con clave secreta).

Si se desea encriptar archivos guardados en un disco rígido con el DES, no es posible cambiar las claves con frecuencia, ello significaría desencriptar y re-encriptar todos los archivos con cada cambio de clave. En cambio hay que tener una clave DES maestra con la cual encriptar la lista de claves DES utilizadas para encriptar los archivos, luego se podrá cambiar la clave maestra con frecuencia y poco esfuerzo.

Una técnica poderosa para mejorar la seguridad del DES radica en el encriptado triple, es decir, encriptar cada bloque de mensaje bajo tres claves DES diferentes. El encriptado triple se considera equivalente al doble del tamaño de la clave de DES, a 112 bits, y debe evitar que un oponente capaz de buscar una sola clave pueda desencriptar mensajes [Merkle and Hellman, 1981]. Por supuesto, utilizar el encriptado triple lleva el triple de tiempo que un encriptado simple.

Referencias

- [Diffie and Hellman, 1977] Diffie W. and Hellman M. E. Exhaustive Criptanalysis of the NBS Data Encryption Standard. Computer, 10:74 – 84, 1977.

- [Matsui, 1993] Matsui, M. Linear cryptanalysis method for DES cipher. In Advances in Cryptology – Eurocrypt'93, Springer-Verlag, Berlin, 1993.

- [Merkle and Hellman, 1981] Merkle R. C. and Hellman M. E. On the security of multiple encryption. Communications of the ACM, 24:465-467, July 1981.

- [Wiener, 1993] Wiener M. J. Efficient DES key search. August 20, 1993. Presented at Crypto'93 rump session.

IV. - CRIPTOGRAFIA CON CLAVE PUBLICA

La criptografía con clave pública fue inventada por Whitfield Diffie y Martin Hellman en 1976 con el propósito de resolver el problema de la administración de claves. En el nuevo sistema, cada persona obtiene un par de claves, llamadas clave pública y clave privada.

La *clave pública* de cada persona se publica y la *privada* se mantiene en secreto. La necesidad del remitente y del receptor de compartir la misma clave queda eliminada: las comunicaciones sólo necesitan de la clave pública y entonces la clave privada no se transmite ni se comparte. Ya no es necesario confiar en los canales de comunicación, corriendo el riesgo de que alguien esté escuchando en la línea telefónica o de que se viole el secreto de la clave privada. Cualquier persona puede enviar un mensaje confidencial con solo utilizar la clave pública, pero el mensaje solo puede desencriptarse con la clave privada que posee únicamente el receptor. Más aún, la criptografía con clave pública puede utilizarse tanto para la autenticación (firmas digitales) como para mantener la privacidad (encriptado).

El *encriptado* de mensajes se utiliza para mantener la privacidad entre partes, y se realiza encriptando el mensaje a enviar con la clave pública del receptor. Por otro lado, el receptor podrá leer el mensaje, y sólo él podrá hacerlo, desencriptando el mensaje con su clave privada.

El *autenticado* permite realizar transferencia de datos en forma confidencial y se realiza enviando la firma y el mensaje original al receptor. Llamaremos firma al mensaje encriptado utilizando la clave privada del emisor. Para poder leer y verificar la autenticidad de la firma, el receptor desencripta la firma con la clave pública del emisor y comprueba que el mensaje obtenido coincida con el enviado.

RIVEST SHAMIR ADLEMAN (RSA)

IV. 1 - Introducción

El trabajo pionero de Diffie y Hellman [DIFF76b] introdujo un nuevo enfoque a la criptografía y desafió a la gente que trabajaba en criptografía a desarrollar un algoritmo criptográfico que cumpla los requerimientos del sistema de clave pública. Una de las primeras respuestas a este desafío fue desarrollada en 1977 por Ron Rivest, Adi Shamir y Leonard Adleman en el MIT, y fue publicado en 1978 [RIVE78]. El sistema Rivest-Shamir-Adleman (RSA) ha reinado desde entonces como la única implementación de encriptación por clave pública desarrollada y ampliamente aceptada.

IV. 2 - Descripción del Algoritmo RSA

El sistema RSA se utiliza tanto para proveer autenticación (firmas digitales) como privacidad (encriptación).

El sistema RSA para encriptado funciona del siguiente modo: supongamos que Alicia quiere enviarle un mensaje privado, *M*, a Juan. Alicia crea el texto cifrado *C*, de modo tal que $C = M^E \bmod N$, donde *E* y *N* son la clave pública de Juan. Para desencriptarlo, Juan recupera el mensaje original computando $M = C^D \bmod N$. La relación entre *E* y *D* asegura que Juan recupere el mensaje correctamente. Como solo Juan conoce el número *D*, solo él puede desencriptar el mensaje.

El sistema RSA de autenticación funciona del siguiente modo: supongamos que Alicia quiere enviarle a Juan un documento *M* firmado. Alicia crea una firma digital *S* tal que $S=M^D \bmod N$, donde *D* y *N* constituyen la clave privada de Alicia. Ella envía el documento

M y la firma *S* a Juan. Para verificar la firma, Juan debe controlar que el mensaje *S^E* mod *N*, donde *E* y *N* constituyen la clave pública de Alicia, sea igual al mensaje *M*.

De este modo el encriptado y la autenticación se producen sin compartir ninguna clave secreta: cada persona utiliza sólo las claves públicas de otros usuarios y sus propias claves privadas. Cualquiera puede enviar un mensaje encriptado o verificar un documento firmado utilizando sólo claves públicas, pero solamente alguien que posee la clave privada correcta puede desencriptar o firmar el mensaje.

El esquema desarrollado por Rivest, Shamir y Adleman hace uso de una expresión con exponenciales. El texto es encriptado en bloques, con cada bloque teniendo un valor binario menor que algún número *n*.

En RSA, cada usuario tiene una clave pública $P = (c, n)$ y una privada $S = (d, n)$, donde c, d y n son enteros. Para generar las claves P y S, un usuario debe proceder de la siguiente manera:

paso 1: Generar 2 números primos grandes distintos p y q, del orden de 10^{100} cada uno

paso 2: Calcular $n = p*q$ y $\phi(n) = (p-1)*(q-1)$

paso 3: Generar un entero c en el intervalo $(1, \phi(n))$, tal que mcd $(c, \phi(n)) = 1$

paso 4: Determinar d a partir de la ecuación $c*d \bmod \phi(n) = 1$

paso 5: Publicar $P = (c, n)$ y mantener secretos d, $\phi(n)$, p y q.

Para encriptar un mensaje usando la clave P, un usuario divide trivialmente el mensaje en bloques de bits que representan enteros m en el intervalo $[0, n)$, es decir que $0 \leq m < n$, y encripta cada m bits usando el procedimiento *expmod* de exponenciación modular rápida (Ver Apéndice A), como se describe:

$$m_1 = P(m) = m^c \bmod n$$

Para desencriptar m_1, el destinatario, que es el único que conoce d también usa el procedimiento *expmod*, de la siguiente manera:

$$S(m_1) = m_1^{\,d} \bmod n$$

En el esquema RSA, tanto quien envía como quien recibe deben conocer el valor de n. Quien envía conoce el valor de c, y sólo el receptor conoce el valor de d.

Es difícil, presumiblemente, obtener la clave privada d a partir de la clave pública (c, n). Sin embargo, si se pudiera descomponer n en p ó q, se podría obtener la clave privada d. Por lo tanto, toda la seguridad del sistema RSA se basa en el concepto de que la descomposición en factores primos es computacionalmente compleja.

Análisis de los Pasos para la Generación de las Claves

La generación de los números primos p y q en el *paso 1*, la analizaremos más adelante (Capítulo IV.5).

El *paso 2* no requiere de un análisis particular, toda vez que consiste en la obtención de los valores de n y $\phi(n)$ a partir de los números p y q obtenidos en el paso 1.

La generación de c en el *paso 3* puede ser realizada utilizando generadores de números pseudo-aleatorios [KNUT2] e iterando:

$$c \leftarrow c \,/\, \text{mcd}\,(c, \phi(n))$$

mientras el mcd $(c, \phi(n))$ sea distinto de 1.

La determinación de mcd $(c, \phi(n))$ es efectuada a través del Algoritmo de Euclides para el cálculo del máximo común divisor (Ver Apéndice A).

El *paso 4* resuelve la congruencia del módulo $\phi(n)$ que determina d. Para ello, utilizamos una extensión del Algoritmo de Euclides [KNUT1] (Ver Apéndice A).

Ejemplo :

Las claves fueron generadas de la siguiente manera:

1. Se eligen 2 números primos, $p = 7$ y $q = 17$.
2. Se calcula $n = pq = 7 \times 17 = 119$
3. Se calcula $\phi(n) = (p - 1)(q - 1) = 96$
4. Se elige c de modo que sea coprimo con $\phi(n) = 96$ y menor que éste; en este caso $c=5$.
5. Se determina d tal que $d*c=1 \bmod 96$ y $d < 96$.

   El valor correcto es $d = 77$, porque $77 \times 5 = 385 = 4 \times 96 + 1$.

Las claves resultantes son: clave pública $P = (5, 119)$ y clave privada $S = (77, 119)$. Para encriptar un texto de entrada $M = 19$, el valor 19 es elevado a la 5ta potencia obteniendo como resultado 2476099. Se divide por 119 y el resto es 66. Así, $19^5 \equiv 66 \bmod 119$, y el texto encriptado es 66. Para desencriptarlo, se determina que $66^{77} \equiv 19 \bmod 119$.

IV. 3 - Relación entre las Claves Pública y Privada

Vamos a verificar que $S*P$ es una identidad, esto es que:

$$S(P(m)) = m \qquad 0 \leq m < n$$

y por lo tanto, el destinatario desencripta los mensajes correctamente. Para esto, observe inicialmente que :

$$S(P(m)) = (m^c \bmod n)^d \bmod n = m^{c*d} \bmod n \qquad (1)$$

Para esta demostración debemos tener presente un importante teorema de la teoría de números como es el Teorema de Fermat.

Teorema de Fermat: Sea p un número primo, m un entero no múltiplo de p. Entonces, para todo natural r, $m^r \bmod p = m^{r \bmod (p-1)} \bmod p$.

En nuestro caso, por la definición de d, $c*d \bmod \phi(n) = 1$, y por lo tanto:

$$c*d = 1 + k\,(p-1)*(q-1)$$

para algún entero k.

Asimismo,

$$c*d \bmod (p-1) = 1$$

Si m no es múltiplo de p, entonces:

$$m^{c*d} \bmod p = m^{c*d \bmod (p-1)} \bmod p = m \bmod p$$

Si m fuera múltiplo de p, m^{c*d} también lo será. Por lo tanto cualquiera que sea m, tenemos:

$$m^{c*d} \bmod p = m \bmod p$$

Asimismo,

$$m^{c*d} = m + s*p$$

para algún entero s. Análogamente:

$$m^{c*d} = m + t*q$$

para algún entero t y por lo tanto:

$$m^{c*d} - m = s*p = t*q$$

Además, p y q son primos distintos, son primos entre sí y como p divide a $s*p$, p divide también a $t*q$, luego p divide a t. O sea:

$$m^{c*d} - m = w*p*q$$

para algún w entero. Luego, por la definición de n tenemos que :

$$m^{c*d} = m + w*n$$

y por lo tanto

$$m^{c*d} \bmod n = m \bmod n = m$$

Por lo realizado y (1), tenemos que:

$$S(P(m)) = m$$

Por lo que queda demostrado la identidad planteada precedentemente.

IV. 4 - Generación de las Claves Pública y Privada

Antes de la aplicación del criptosistema de clave pública, cada usuario debe generar un par de claves. Esto involucra las siguientes tareas:

1 Determinar 2 números primos, p y q.

Debido a que el valor $n = p*q$ podría ser conocido por cualquier oponente potencial y dado que los números p y q podrían ser descubiertos por métodos exhaustivos, estos números primos deben ser elegidos desde un conjunto suficientemente grande para evitar su publicidad. Por otro lado, el método usado para encontrar primos grandes debe ser razonablemente eficiente.

El procedimiento para obtener un número primo es el siguiente:

i) Seleccionar un entero impar n en forma aleatoria (Por ejemplo, utilizando un generador de números pseudoaleatorios).

ii) Seleccionar un entero a< n en forma aleatoria.

iii) Realizar el testeo de primalidad probabilístico (Miller-Rabin). En caso que el testeo falle con n, se deberá rechazar el valor de n y volver al *paso i)*

iv) En caso que n haya pasado el testeo exitosamente un número suficiente de veces, se acepta el valor de n, sino se pasa al *paso ii).*

2 Seleccionar c o d y calcular el otro (exponente público y privado respectivamente).

Habiendo determinado los números primos p y q, el proceso de generación de claves es completado seleccionando un valor de d y calculando c o alternativamente, seleccionando un valor de c y calculando d.

Necesitamos seleccionar un d tal que mcd $(\phi(n), d) = 1$ y luego calculamos:

$$c = d^{-1} \bmod \phi(n)$$

Afortunadamente existe un algoritmo simple que al mismo tiempo calcula el máximo común divisor (mcd) de dos números enteros y si el mcd es 1, determina el inverso de uno de los enteros módulo el otro. El algoritmo al que nos referimos es una extensión del Algoritmo de Euclides.

Por lo tanto, el procedimiento consistiría en generar una serie de números aleatorios, testeando cada uno de ellos contra $\phi(n)$ hasta que un número relativamente primo a $\phi(n)$ sea encontrado.

¿ Cuántos números al azar debemos testear para encontrar un número relativamente primo a $\phi(n)$?

Es fácilmente demostrable que la probabilidad de que dos números al azar sean relativamente primos es de aproximadamente 60/100; por lo tanto, muy pocos testeos deberíamos necesitar para encontrar un entero apropiado.

Extensión del Algoritmo de Euclides

Si mcd$(d, f) = 1$, entonces d tiene un multiplicador inverso módulo f. Esto es, para enteros positivos $d < f$, existe un $d^{-1} < f$ tal que $d*d^{-1} = 1$ (módulo f). El Algoritmo de Euclides puede ser extendido para encontrar el mcd(d, f) y si el máximo común divisor es 1, retorna el multiplicador inverso de d. (Ver Apéndice A).

IV. 5 - Generación de los números primos *p* y *q*

No se conoce ningún algoritmo determinístico eficiente para comprobar la primalidad de un número entero. No obstante, existen algoritmos eficientes probabilísticos para tal fin. Estos algoritmos son suficientes para generar *p* y *q*, conforme veremos en seguida.

Para generar un número primo *p* en el intervalo $[10^{100}, 10^{101}]$ se puede proceder de la siguiente manera: se genera aleatoriamente un número entero ***b*** impar en el intervalo dado, con distribución uniforme, y se verifica si ***b*** es primo. Si **b** no fuera primo, se genera otro ***b*** y así se repite este proceso hasta obtener un ***b*** primo.

¿ Cuántos testeos de primalidad precisaremos hacer, hasta obtener un número ***b*** primo ?

El teorema de los números primos afirma que una función $\Pi(x)$, igual al número de primos menores que el natural x, tiene el siguiente comportamiento:

$$Lim \xrightarrow[x \to \infty]{} \frac{\Pi(x)}{(x/\ln x)} = 1$$

Así, podemos estimar que:

$$\Pi(10^{100}) = \frac{10^{100}}{\ln 10^{100}}$$

y

$$\Pi(10^{101}) = \frac{10^{101}}{\ln 10^{101}}$$

Luego la probabilidad de que el número b impar en el intervalo sea primo es:

$$\frac{\Pi(10^{101}) - \Pi(10^{100})}{5 \times (10^{101} - 10^{100})} = \frac{2 \times 10^{-2}}{\ln 10} = \frac{1}{115}$$

Por lo tanto, deberán ser generados cerca de 115 números ***b*** impares antes que un número primo sea encontrado.

Testeos de Primalidad

No existe todavía una forma simple y eficiente para determinar si un número grande es primo. Vamos a describir un método que presenta una atractiva aproximación. Primero, necesitamos obtener algunos resultados.

Si p es un primo impar, entonces la ecuación $x^2 \equiv 1$ (módulo p) tiene sólo 2 soluciones, que son $x \equiv 1$ y $x \equiv -1$. Esto es fácilmente demostrable :

$$x^2 - 1 \equiv 0 \text{ (módulo } p)$$

$$(x + 1)(x - 1) \equiv 0 \text{ (módulo } p)$$

Por las reglas de la aritmética modular, la última ecuación requiere que p divida a $(x + 1)$ o que p divida a $(x - 1)$, o a ambos. Supongamos que p divide a ambos $(x + 1)$ y $(x - 1)$. Luego podemos decir que :

$$(x + 1) = k*p \quad \text{y} \quad (x - 1) = j*p$$

para algún entero k y j.

Restando las dos ecuaciones obtenemos $2 = (k - j)*p$. Esta ecuación puede ser verdadera sólo para $p=2$. Por los términos del teorema sólo conciernen los primos impares. Por lo tanto, para la ecuación dada, $p/(x + 1)$ ó $p/(x - 1)$, pero no a ambos.

Supongamos que $p/(x - 1)$, entonces $x - 1 = k*p$ para algún k. Por lo tanto, $x \equiv 1$ (módulo p). Razonando en forma similar, arribamos a la otra solución, $x \equiv -1$ (módulo p).

El teorema puede ser planteado desde otro punto : si existen otras soluciones para $x^2 \equiv 1$ (módulo p) que no son 1 y -1, entonces n no es un número primo.

Método de Solovay y Strassen

Vamos a describir el método para el testeo de primalidad de *Solovay* y *Strassen*. Supongamos que se desea determinar la primalidad de un número natural dado b (impar). Sea a un número entero que pertenece al intervalo $[1, b)$, es decir que $1 \leq a < b$. Si b fuera primo, entonces vale la relación para b impar y a primo relativo de b, es decir que a y b son coprimos :

$$\text{mcd}\,(a, b) = 1$$

y

$$J(a, b) \bmod b = a^{(b-1)/2} \bmod b \qquad (1)$$

donde $J(a, b)$ es el símbolo de Jacobi (Ver Apéndice A).

Por otro lado, si b no fuera primo, entonces (1) vale para los valores estrictamente menores a la mitad de los valores en el intervalo $[1, b)$, Así, podemos generar independientemente, digamos 100 valores en el intervalo $[1, b)$ con distribución uniforme. En este caso, la probabilidad de valer la igualdad (1) para todos los 100 valores de a distribuidos uniformemente, es menor que $2^{-100} \approx 10^{-30}$.

Asimismo, si b fuera primo, la ecuación (1) vale para cada uno de los 100 valores de a generados.

En definitiva, el algoritmo consiste en verificar (1) para los 100 valores de a. Si la ecuación (1) vale para todos los valores, el algoritmo decide que b es primo (con probabilidad de error menor que 10^{-30}); si la igualdad falla para por lo menos uno de los valores de **a**, el algoritmo decide que b es compuesto (con probabilidad de error nula).

La ecuación (1) es testeada eficientemente con el auxilio del procedimiento *expmod* (exponenciación modular rápido).

El algoritmo de Jacobi y el algoritmo del máximo común divisor son extremadamente eficientes y operan un número de operaciones aritméticas proporcional a la suma de los números de bits de ***a*** y ***b***.

Método de Miller y Rabin

Veremos ahora un algoritmo desarrollado por *Miller* y *Rabin* [MILL75,RABI80] para testear si un número es primo. El corazón del algoritmo, llamado WITNESS, es definido como sigue :

WITNESS (***a***, ***n***)

1. Sea $b_k\ b_{k-1} \ldots\ b_0$ la representación binaria de (n-1)
2. d = 1
3. For i = k Down to 0
4. x = d
5. $d = d^2 \bmod n$
6. if d = 1 y x <> 1 y x <> (n-1)
7. then Return TRUE
8. if b_i = 1
9. then d = (d * a) mod n
10. if d <> 1
11. then Return TRUE
12. Return FALSE

Las entradas de WITNESS son los números ***n***, a ser testeado y algún entero ***a*** menor que ***n***. El propósito es testear si ***n*** es primo. Si WITNESS retorna TRUE, entonces ***n*** es definitivamente no primo. Si WITNESS retorna FALSE, entonces ***n*** puede ser primo.

En las líneas 3 a 9 del algoritmo WITNESS, se calcula d como el valor $a^{n-1} \bmod n$. Sabemos por el teorema de Fermat que $a^{n-1} \equiv 1 \pmod{n}$, si n es primo. Por lo tanto, si el resultado final para d no es igual a 1, tenemos que n no es primo y retorna TRUE.

Consideremos ahora el testeo de la línea 6. Dado que $(n - 1) \equiv -1 \pmod{n}$, esta línea testea si $x^2 \equiv 1$ (modulo n) vale para otra raíz que no es 1 ni -1. Por el teorema visto anteriormente, esta condición vale solo si x no es primo. Por lo tanto, si el testeo resulta exitoso, WITNESS retorna TRUE.

Ejemplo :

Sea $n = 561$ y $a = 7$. WITNESS descubre una raíz cuadrada que no es 1 ni -1 en el último paso dado que $a^{280} \equiv 67$ (modulo 561) y $a^{560} \equiv 1$ (modulo 561). En este punto, WITNESS retorna TRUE.

Por lo tanto, si WITNESS retorna TRUE, el número n no es primo. Podemos mostrar que dado un número impar n que no es primo y un entero a elegido aleatoriamente, tal que $a < n$, la probabilidad que WITNESS retorne FALSE (no detectando que n no es primo) es menor que 0,5. Esto nos da la base para determinar si un entero impar n es primo con un grado razonable de confianza.

El procedimiento es el siguiente : repetimos la invocación de WITNESS (a, n), usando valores para a elegidos aleatoriamente (randomly). Si en algún punto WITNESS retorna TRUE, entonces n resulta ser "no primo". Si WITNESS retorna FALSE s veces en una sesión, entonces la probabilidad que n sea primo es al menos $1 - 2^{-s}$.

Por lo tanto, para un valor suficientemente grande de s, podemos tener la confianza que n es primo.

IV. 6 - Aspectos Computacionales

Analizaremos ahora la complejidad de los cómputos necesarios para aplicar el RSA. Hay dos aspectos a considerar: la generación de claves y la encriptación/desencriptación.

Tanto la encriptación como la desencriptacion en RSA involucran elevar un entero a una potencia entera, módulo n. Si la exponenciación es realizada bajo estos enteros y luego es reducida módulo n, los valores intermedios podrán ser extremadamente grandes. Afortunadamente, podemos hacer uso de una propiedad de la aritmética modular:

$$[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$$

De esta manera podemos reducir los resultados intermedios módulo n. Esto hace que el cálculo sea más práctico.

Otra consideración es la eficiencia de la exponenciación, desde que con RSA estamos trabajando con exponentes potencialmente grandes. Para ver como podemos incrementar la eficiencia, consideremos que queremos calcular x^{16}. La forma directa para el cálculo de x^{16} requiere 15 multiplicaciones:

$$x^{16} = x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x \times x$$

Sin embargo, podemos obtener el mismo resultado con sólo cuatro multiplicaciones si tomamos el cuadrado de cada resultado parcial, obteniendo sucesivamente: x^2, x^4, x^8, x^{16}.

De forma más general, supongamos que queremos encontrar el valor a^m, con a y m enteros positivos. Si expresamos m como un número binario $b_k b_{k-1} \ldots b_0$, tenemos:

$$m = \sum_{b_i \neq 0} 2^i$$

Así:

$$a^m = a^{\left(\sum_{b_i \neq 0} 2^i\right)} = \prod_{b_i \neq 0} a^{(2^i)}$$

$$a^m \bmod n = \left[\prod_{b_i \neq 0} a^{(2^i)}\right] \bmod n = \prod_{b_i \neq 0} \left[a^{(2^i)} \bmod n\right]$$

Podemos entonces desarrollar el siguiente algoritmo para calcular a^b mod n:

```
c=0; d=1
for i=k downto 0
  do c=(2*x)
     d=(d*d)mod n
     if bi=1 then
       c=c+1
       d=(d*a)mod n
return d
```

Nótese que la variable *c* no es necesaria, ésta es incluida con el propósito de explicar el procedimiento. El valor final de *c* es el valor del exponente.

IV. 7 - Criptoanálisis

La forma más simple de ataque a un mensaje consiste en adivinar el texto plano. El atacante ve un texto cifrado, infiere que el mensaje dice "Atacar al amanecer" y encripta este mensaje inferido con la clave pública del destinatario. Por comparación con el texto cifrado original, el atacante sabe si su mensaje inferido ha sido el enviado. El ataque puede ser desbaratado si se agregan algunos bits al azar al mensaje. Otro ataque al mensaje puede ocurrir si alguien envía el mismo mensaje *m* a otros tres usuarios, quienes a su vez tienen el exponente público e = 3. El atacante que sabe esto y ve los tres mensajes podrá recobrar el mensaje *m*. Existen algunos ataques a textos cifrados "elegidos" en los cuales el atacante crea un texto cifrado y logra ver el correspondiente texto plano, quizás engañando a un usuario legítimo para que desencripte un mensaje falso.

Desde el punto de vista del criptoanalisis del algoritmo RSA, se pueden identificar cuatro métodos para intentar conocer la clave privada y así quebrar el sistema:

- *Fuerza bruta*: probar todas las claves privadas posibles
- *Descomponer* n en sus 2 factores primos. Esto permite calcular $\phi(n) = (p-1) * (q-1)$, lo que permite luego calcular d.
- *Determinar* $\phi(n)$ *directamente*, sin determinar p y q antes. Esto, nuevamente, permite la determinación de d.
- *Determinar* d *directamente*, sin determinar $\phi(n)$ antes.

La defensa contra el método de fuerza bruta es la misma que para otros sistemas criptográficos, esto es, usando un espacio de claves grande. Así, cuanto mayor sea el número de bits que tengan c y d, mejor. Sin embargo, debido a que los cálculos requeridos para la generación de claves y para la encriptación y desencriptación son complejos, cuanto mayor sea el tamaño de la clave, más lento será el sistema.

Respecto del segundo método, no existe en la actualidad un algoritmo razonable que pueda descomponer el producto de dos números primos para valores muy grandes (varios cientos de dígitos decimales). Hasta que se descubran mejores algoritmos, una magnitud de n de 100 o 200 dígitos es seguro. Por ejemplo, a un nivel de 10^{12} operaciones por segundo, tomaría del orden de 100 años descomponer un número de 200 dígitos con alguno de los algoritmos existentes. Para una implementación eficiente del algoritmo RSA, una magnitud de 154 dígitos, que puede ser manejada en 512 bits es atractiva, mientras que un número de 200 dígitos requiere alrededor de 660 bits y es poco práctico para implementar.

Se puede argumentar que un sistema será quebrado con el conocimiento de $\phi(n)$, sin necesidad de conocer p y q, de hecho, dados c y $\phi(n)$ el exponente secreto d puede ser calculado. No obstante, el conocimiento de $\phi(n)$ y de n, permite descomponer n rápidamente:

$$n - \phi(n) + 1 = n - (p - 1)(q - 1) + 1 = p + q$$

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

y por lo tanto p y q son raíces de la ecuación:

$$x^2 - (n - \phi(n) + 1) * x + n = 0$$

Lo que vimos hasta ahora permite concluir que descomponer n y determinar $\phi(n)$ son problemas computacionalmente equivalentes y que el éxito de encontrar cualquiera de ellos implica quebrar el RSA por la determinación del exponente privado d.

Con los algoritmos existentes, los métodos para determinar $\phi(n)$ o d, resultan tan lentos como el problema de factorización. Así, podemos evaluar la performance de la factorización como parámetro para evaluar la seguridad del RSA.

Además de requerir que n sea del orden de 10^{150} a 10^{200}, algunas otras restricciones han sido propuestas por investigadores. Para evitar valores de n que sean fácilmente factorizables, los inventores del algoritmo sugieren las siguientes restricciones para p y q.

1. Los números p y q deben diferir sólo en unos pocos dígitos en su largo. Así, p y q deben ser del orden de 10^{75} a 10^{100}. Caso contrario p y q figuraran muy próximos a raíz de n.
2. Tanto $(p - 1)$ como $(q - 1)$ deben contener un factor primo grande.
3. El mcd$(p - 1, q - 1)$ debe ser pequeño.

De este modo, el módulo n es más difícil de descomponer en sus factores que cuando uno de los números primos que componen el módulo es pequeño. Si se elige utilizar un módulo de 512 bits, los números primos deberían tener una longitud de 256 bits.

El tamaño de un módulo RSA depende de las propias necesidades de seguridad. Cuanto más largo sea el módulo, mayor es la seguridad pero también más lentas son las operaciones RSA. Se puede elegir un largo de módulo considerando, primero, las necesidades de seguridad, es decir el valor de la información protegida y, segundo, cuánto tiempo necesita estar protegida y cuán poderoso puede ser el potencial enemigo. También es posible que una

clave de mayor tamaño permita que un documento firmado digitalmente sea válido por un lapso mayor de tiempo.

Además, se ha demostrado que, si $c < \phi(n)$ y $d < n^{1/4}$, entonces d puede ser determinado fácilmente [WIEN90].

Referencias

- [DIFF76b] Diffie, W., and Hellman, M. "New Directions in Cryptography". IEEE *Transactions on Information Theory*, Noviembre 1976.

- [KNUT1] D. E. Knuth, "The Art of Computer Programming". *Volumen 1: Fundamental Algorithms*. Reading, MA: Addison-Wesley, 1973.

- [KNUT2] D. E. Knuth, "The Art of Computer Programming". *Volumen 2: Seminumerical Algorithms*. Reading, MA: Addison-Wesley, 1981.

- [MILL75] Miller, G. "Riemann's Hypothesis and Test for Primality". *Proceedings of the Seventh Annual ACM Symposium on the Theory of Computing*, Mayo 1975.

- [RABI 80] Rabin, M. "Probabilistic Algorithms for Primality Testing". *Journal of Number Theory*, Diciembre 1980.

- [RIVE78] Rivest R.; Shamir, A and Adleman, L. "A method for Obtaining Digital Signatures and Public Key Cryptosystems". *Communications of the ACM*, Febrero 1978.

- [WIEN90] Wiener, M. "Cryptanalysis of Short RSA Secret Exponents". IEEE *Transactions on Information Theory*, vol IT-36, 1990.

Apéndice A

RSA – Descripción de los Procedimientos

Procedimiento *Euclides* (*a*, *b*)

```
Begin
     x = a
     y = b
     repetir
          r = x mod y
          x = y
          y = r
     hasta que (r = 0)
     devolver (x)
end
```

Procedimiento *Euclides-Recursivo* (*a* ,*b* ,*c*)

```
Begin
     r = b mod a
     Si (r = 0 )
           entonces
            devolver c div a mod b
          sino
            devolver ( Euclides-Recursivo ( r, a, -c) * b + c ) div a mod b
     end
end
```

Procedimiento *Euclides-Iterativo* (*a*, *b*, *c*)

```
Begin
     x = b
     y = a
     u = 0
     v = 1
     w = 1
     z = 0
     repetir { x = u*a + v*b ; y = w*a + z*b }
          q = x div y
          x = x - q*y
          u = u - q*w
          v = v - q*z
          x ↔ y
          u ↔ w
          v ↔ z
     hasta que (y = 0)    { x = mcd (a, b) ; a*u mod b = x }
     devolver (u * (c div x)) mod b
end
```

Procedimiento ***expmod*** **(*a*, *x*, *n*)**

{Calcula a^x mod n rápidamente. Supongamos que $n > 0$ y $x >= 0$ }

```
Begin
   r = 1
   y = x
   c = a mod n
   En cuanto (y > 0) hacer
      Si (y es impar)
         Entonces r = r*c mod n
      y = y div 2      {siendo div la división entera}
      c = c^2 mod n
   end
   Devolver r
end
```

Procedimiento ***Jacobi*** **(*a*, *b*)**

{*b* es un número impar, *a* es un entero tal que $1 <= a < b$ y el mcd(a, b) = 1 }

```
Begin
   r =1
   x = a; y = b
   En cuanto (x > 1) hacer { mcd(a, b) = mcd(x, y) y y es impar }
      Si (x mod 2 = 0 )
         Entonces
                Begin
                   x = x div 2
                   Si (y mod 8 = 3) or  (y mod 8 = 5)
                      Entonces r = -r
                end
         Sino
                Begin
                   Si (x mod 4 =- 3) and (y mod 4 = 3)
                      Entonces r = -r
                   y = y mod x
                   temp = x
                   x = y
                   y = temp
                end
      end
      Devolver r
end
```

Procedimiento *mcd* (a, b)

{Siendo b un número natural impar y a un entero en el intervalo $1 \leq a < b$}

```
Begin
   x = a; y = b
   Repetir
      Si (x mod 2 = 0)
         Entonces   x = x div 2
         Sino
              Begin
                 y = y mod x
                 aux = x
                 x = y
                 y = aux
              end
   Hasta que (x = 0)
   Devolver y
End
```

V. - ADMINISTRACION DE CLAVES EN UN SISTEMA CRIPTOGRAFICO

La administración de claves en un sistema criptográfico involucra la generación, distribución y protección necesaria de las claves, para que en un sistema secreto de comunicaciones tanto el emisor como el receptor tengan garantizada la seguridad.

En un sistema multi-usuario, siendo n el número de usuarios, las conexiones entre pares de usuarios crece en el orden de $(n^2-n)/2$.

La mayoría de los ataques a los sistemas con clave pública apuntan a los niveles de administración de claves más que al algoritmo criptográfico en sí mismo. Los usuarios deben poder obtener de manera segura un par de claves que se adecue a sus necesidades específicas de seguridad y eficiencia. Deben contar con un medio para conocer las claves públicas de otros usuarios y publicar las propias. Deben tener confianza en la legitimidad de las claves ajenas. De otro modo, un intruso puede cambiar las claves públicas publicadas en el directorio o hacerse pasar por otro usuario.

En una estructura jerarquizada arborescente, el problema de la distribución de claves puede resolverse a través de la cadena de mando, pero aún en este punto hay serios impedimentos en el uso de la criptografía. En términos generales, la utilización de alguno de los esquemas asimétricos facilita la distribución de claves.

V. 1 - Solución mediante Jerarquía de Claves

Propuesta por Everton [Everton,78] el manejo de claves está basado en un principio simple según el cual, cuando una clave no puede ser físicamente protegida, se debe encriptar bajo otra clave de orden superior.

Este esquema se basa en el establecimiento de una jerarquía de claves en cuya raíz se encuentran las claves maestras, que son almacenadas en los dispositivos de encriptación y utilizadas para proteger el nivel siguiente inferior de claves o claves sub-maestras. Estas últimas a su vez son almacenadas en los nodos, si se trata de una red y son utilizadas para proteger el nivel inmediatamente inferior de claves, que en un entorno transaccional suelen denominarse claves de sesión.

Las claves de sesión son las de más bajo nivel siendo utilizadas únicamente para proteger datos y permaneciendo activas solamente mientras dura la sesión de comunicaciones.

Las claves maestra y sub-maestra son generadas por un responsable de seguridad con la ayuda eventual de una computadora dedicada a esta tarea. Las claves maestras permanecerán en forma de texto plano ya que no hay claves de nivel superior a éstas.

Everton sugiere que las claves maestras y las claves sub-maestras encriptadas se transporten a los nodos por un medio seguro, probablemente no por la red, entregando el archivo de claves al representante responsable de seguridad en cada uno de los puntos, permaneciendo ocultos en algún sitio seguro como una caja fuerte.

Como puede deducirse de este sistema, se propone el montaje de una organización con responsables de seguridad, de los cuales depende la correcta distribución secreta y restringida de las claves.

La clave maestra de cada nodo es introducida por el correspondiente responsable de seguridad en el dispositivo de encriptación en forma de texto plano. Dicha clave debería destruirse si hay algún fallo de dispositivo, permitiéndose su reentrada.

Las claves de sesión serían producidas en puntos determinados y transmitidas a los nodos al iniciarse cada sesión, estando encriptadas bajo la clave sub-maestra local. Dentro del dispositivo, la clave sería desencriptada a su forma original (texto plano), no existiendo ninguna manera de obtener esta clave fuera del dispositivo. Para la transmisión a los otros nodos, la clave se encriptará utilizando la clave sub-maestra de los nodos de destino.

En conclusión, este procedimiento es transparente al usuario final y excepto el responsable de seguridad en cada nodo, no hay nadie que tenga que manipular las claves. En este esquema organizado jerárquicamente las claves no son públicas sino secretas.

V. 2 - Solución mediante Claves Centralizadas

Una forma alternativa para distribuir las claves secretas es un método híbrido utilizado en mainframes. Esta solución ha sido propuesta por [IBM, 1978] y se basa en la existencia de una clave maestra en el computador central "Host Master Key" (HMK). El criterio de centralización consiste en la generación de claves sólo en este computador central, siendo transferidas a los demás sitios donde se necesitan. La HMK se almacena como texto plano en el dispositivo que efectúa el encriptado del computador central, impidiendo su acceso por partes no autorizadas.

Cada terminal atendido por este computador central necesita su propia clave maestra de terminal que denominaremos "Terminal Master Key" (TMK), almacenada como texto plano en su dispositivo de encriptación. Asimismo, el computador central guarda una copia encriptada de cada TMK.

Respecto a la distribución de claves, la regla general es que las claves que están fuera de un dispositivo de encriptación deben estar siempre en forma de criptograma, es decir encriptadas con otra clave de nivel superior, excepto las HMK y TMK que deben ser transportadas y creadas como texto plano, ya que no hay claves de orden superior que puedan encriptarlas y desencriptarlas, lo que implica que su manejo deberá ser totalmente secreto.

El sistema de encriptación es transparente el usuario final, relevándose por lo tanto la responsabilidad del manejo de las claves de sesión ya que las mismas pueden ser transportadas por la red encriptadas por la TMK.

Se sugiere que la distribución de las claves a más alto nivel sea realizada por medios extraordinariamente seguros.

Análisis de estas dos Alternativas

Las dos soluciones corresponden a un nivel de tecnología y desarrollo teórico de sistemas secretos, en los que no está resuelto claramente el problema de transporte y distribución de las claves de alto nivel.

Las dos soluciones, jerárquica y centralizada, resultan tener una gran similitud, pudiéndose obtener diseños híbridos para aumentar las ventajas y reducir los inconvenientes, tomando aquellos aspectos que interesen en cada método. El control centralizado es un caso degenerado del control jerárquico, el que se plantea en términos más genéricos.

V. 3 - Distribución de clave pública

Se han propuesto muchas técnicas para la distribución de claves públicas. Virtualmente, todas estas propuestas pueden ser agrupadas de la siguiente manera:

- Anuncio público
- Directorio al alcance público
- Autoridad de clave pública
- Certificados de clave pública

Anuncio Público de las claves Públicas

El punto principal de la encriptación de clave pública es que la clave pública es pública. Así, si existe un algoritmo de clave pública ampliamente aceptado como el RSA, cada participante puede enviar su clave pública a cualquier otro participante o publicitarla

ampliamente para que toda la comunidad la conozca. Este enfoque tiene un punto débil y es que cualquiera puede falsear el anuncio público. Así, algún usuario puede pretender ser el usuario A y enviar una clave pública a otro participante o publicarla. Para cuando el usuario A descubre el hecho y alerta a los otros participantes, el falsificador pudo leer todos los mensajes encriptados enviados a A y usar la clave falsificada para autenticación.

Directorio al alcance público

Un mayor grado de seguridad se puede alcanzar manteniendo un directorio dinámico de alcance público de claves públicas. El mantenimiento y la distribución del directorio público deberá ser responsabilidad de una entidad u organización confiable (Figura 1). Este enfoque deberá incluir los siguientes elementos:

1. La autoridad mantiene un directorio con una entrada {nombre, clave pública} para cada participante.
2. Cada participante registra una clave pública con la autoridad del directorio. El registro deberá ser en persona o por alguna forma de comunicación autenticada.
3. Un participante podrá reemplazar la clave existente con una nueva en cualquier momento, ya sea por haber sido usada para una gran cantidad de información o por haber sido comprometida en alguna forma.
4. Periódicamente, la autoridad publicará el directorio entero o las actualizaciones al mismo.
5. Los participantes podrán acceder al directorio en forma electrónica. Para este propósito, se requiere una forma de comunicación autenticada entre la autoridad y el participante.
6. Este enfoque es, sin dudas, más seguro pero aún tiene sus puntos vulnerables. Si alguien logra obtener la clave pública de la autoridad del directorio, podría distribuir claves públicas falsas y así, imposibilitar la recepción de mensajes enviados por las personas cuyas claves fueron falsificadas.

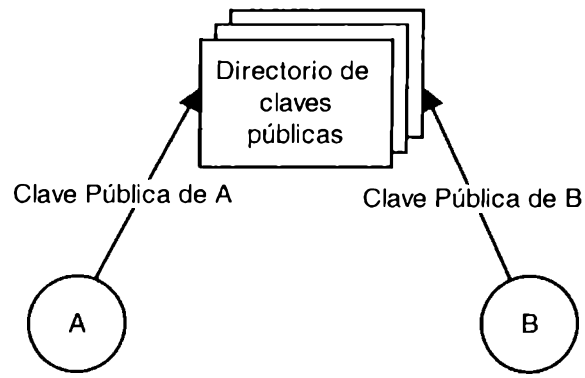


Figura 1 – Publicación de Claves Públicas

Autoridad de claves públicas

Una manera de conseguir mayor seguridad en la distribución de claves públicas es teniendo un mayor control en la forma en que se distribuyen (Figura 2). Para asegurar esto, la forma en que la autoridad distribuye claves debería seguir ciertas pautas de seguridad. Un mecanismo tentativo sería el siguiente:

1. El usuario A le envía a la autoridad de claves públicas un mensaje con información de fecha y hora requiriendo la clave pública de B.
2. La autoridad responde con un mensaje encriptado usando su clave privada, el cual A puede desencriptar a través de la clave pública de la autoridad. El mensaje contiene la siguiente información:
   - La clave pública de B.
   - El pedido original, para ser comparado por A y verificar que éste no fue alterado antes de ser recepcionado por la autoridad.
   - Fecha y hora del envío original, para que A pueda asegurarse de que no se trata de un mensaje antiguo.

3. A guarda la clave pública de B y la usa para encriptar el mensaje para B el cual contiene un identificador de A y un código temporario, el que es utilizado para identificar unívocamente el envío.
4. B obtiene la clave pública de A de la autoridad de la misma manera que A obtuvo la de B.

Hasta este punto, se les han enviado, de una manera segura, las claves públicas a A y B, y podrán realizar el envío protegido. Sin embargo, son deseables dos pasos adicionales:

5. B envía un mensaje a A encriptado con la clave pública de A conteniendo el código temporario de A y uno nuevo generado por B. Dado que B es el único que pudo haber desencriptado el mensaje (3), la presencia del código temporario en este mensaje le asegura a A que está hablando con B.
6. A devuelve el código temporario de B, encriptado usando la clave pública de B, para asegurarle a B que está hablando con A.

Así, se requieren un total de 7 mensajes (el paso 4 implica dos). Sin embargo, los primeros cuatro mensajes no se realizarán muy frecuentemente, dado que A y B pueden guardar la clave pública del otro para un uso futuro. Periódicamente, el usuario deberá requerir la copia actual de la clave pública del otro usuario por si ha habido algún cambio.

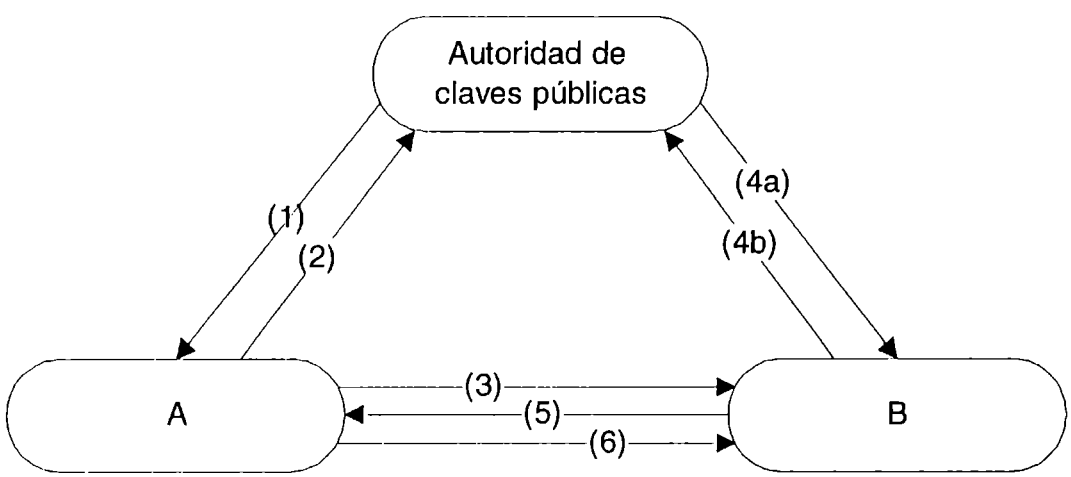


Figura 2 – Escenario de Distribución de Claves Públicas

Certificados de clave pública

El sistema de autoridad de claves es atractivo pero presenta algunas desventajas. La autoridad de clave pública puede ser, en muchos casos, un cuello de botella del sistema, ya que cada usuario debe recurrir a éste cada vez que se quiere contactar con otro usuario. Asimismo, el directorio de nombres y claves públicas mantenido por una autoridad está expuesto a falsificaciones.

Una alternativa a estos esquemas, propuesto por Kohnfelder [KOHN,78], es crear certificados que pueden ser usados por los usuarios para intercambiar claves sin conectarse con una autoridad de claves públicas. Cada certificado contiene la clave pública y otra información, es creada por una autoridad y es entregada a cada usuario con su correspondiente clave pública. Un usuario distribuye su clave enviando el certificado. Los demás usuarios pueden verificar que el certificado fue creado por la autoridad. Para obtener el certificado, cada usuario debe acudir a la autoridad, ya sea en persona o por algún medio de comunicación seguro, con su clave. Al enviársela a otro usuario, este la desencripta con la clave pública de la autoridad, lo que verifica que el certificado proviene de ésta. Así, el usuario obtiene la información del nombre y la clave pública del dueño del certificado y una marca de fecha asegura la validez de los datos (Figura 3).

En este contexto, el ver comprometida una clave pública, es comparable a la pérdida de una tarjeta de crédito. El dueño cancela el número de tarjeta de crédito pero no está a salvo hasta que todos los posibles receptores están al tanto de esta cancelación. Así, la marca temporal sirve como una fecha de expiración. Si un certificado es suficientemente viejo, se asume que ha expirado.

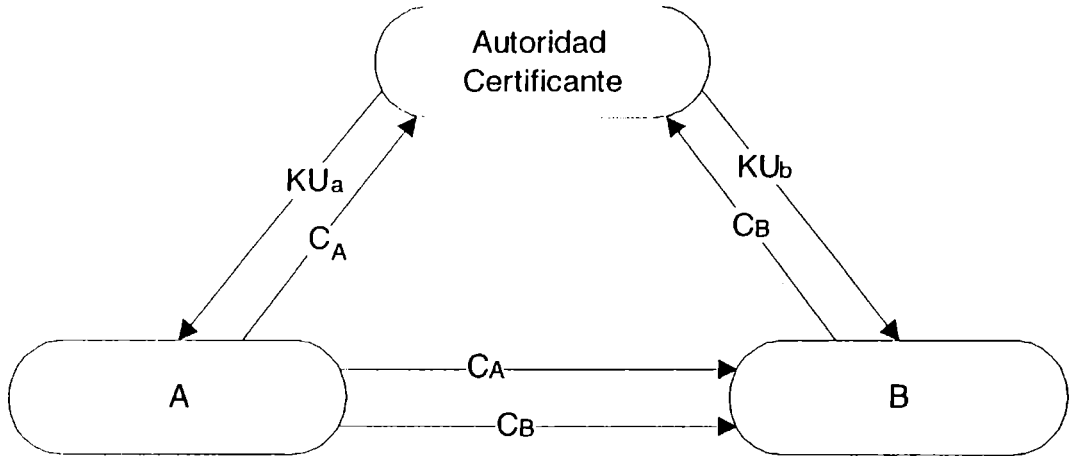


Figura 3 – Intercambio de Certificados de Clave Pública

Los certificados son documentos digitales que atestiguan que una clave pública pertenece a determinado individuo o entidad. Permiten la verificación en caso de reclamos de que una clave pública dada pertenece fehacientemente a una determinada persona. Los certificados ayudan a evitar que alguien utilice una clave falsa haciéndose pasar por otro.

En su forma más simple, los certificados contienen una clave pública y un nombre, la fecha de vencimiento de la clave, el nombre de la autoridad certificante, el número de serie del certificado y quizás otros datos. Esencialmente, contiene la firma digital del que otorga el certificado.

Los certificados los emite una autoridad certificante (AC), que puede ser cualquier administración central de confianza que pueda confirmar las identidades de aquellos a quienes otorga los certificados. Una compañía puede emitir certificados a sus empleados, una universidad a sus estudiantes, una ciudad a sus ciudadanos. Para evitar que se falsifiquen los certificados, la clave pública de la AC deber ser confiable: una AC debe publicar su clave pública o proporcionar un certificado de una autoridad mayor que atestigüe la validez de su clave. Esta solución da origen a jerarquías de Autoridades Certificantes.

La emisión de certificados funciona del siguiente modo. Alicia genera su propio par de claves y envía su clave pública a una AC apropiada, con alguna prueba de su identidad. La AC corrobora la identificación y toma otras medidas necesarias para asegurarse de que el pedido haya procedido de Alicia y luego le envía a Alicia un certificado que atestigua que la clave le pertenece, junto con la jerarquía de certificados que verifican la clave pública de esa AC. Alicia puede presentar esta cadena de certificados cuando desee demostrar la legitimidad de su clave pública.

V. 4 - Distribución de claves secretas con claves públicas

Una vez que las claves públicas han sido distribuidas o son accesibles, es posible realizar comunicaciones seguras. Sin embargo, pocos usuarios desearán hacer uso exclusivo de la encriptación de clave pública para sus comunicaciones debido a la baja velocidad de transferencia que se obtiene. Es por eso que la encriptación con clave pública es utilizada como medio de distribución de claves secretas para ser usadas en encriptación convencional.

Distribución simple de Clave secreta

En un modelo simple propuesto por Merkle [MERK,79], si un usuario A se desea comunicar con B, debe seguir los siguientes pasos:

1. A genera un par de claves pública y privada y envía un mensaje a B conteniendo su clave pública y un identificador.
2. B genera una clave secreta y se la envía a A, encriptada con la clave pública de A.
3. A recupera la clave secreta de B con su clave privada.
4. A y B desechan las claves generadas en el paso 1.

Ahora A y B pueden comunicarse de una manera segura usando encriptación convencional y la clave secreta generada en 2). Además de su simplicidad, éste es un protocolo muy atractivo por su gran seguridad ya que es sólo vulnerable a un ataque activo. Si alguien tiene control del canal de comunicación que esta siendo utilizado, puede utilizar sus propias claves en el paso 2 y enviárselas a B. Luego, intercepta la clave secreta, la que recupera con su clave pública, y la envía luego a A, usando la clave pública de éste, por lo que ni A ni B detectarán la intromisión, pero el intruso conocerá la clave secreta que estos utilizarán para comunicarse. Es por esto que este protocolo es muy útil en ambientes en los que la única amenaza son las escuchas.

Distribución de Clave secreta con confidencialidad y autenticación

Este protocolo provee seguridad contra ataques activos y pasivos. Supondremos que A y B han intercambiado claves públicas por alguno de los métodos descriptos antes. Luego, los siguientes pasos se deben seguir:

1. A envía un mensaje a B encriptado con la clave pública de B conteniendo un identificador de A y un código temporario N_1 que es usado sólo para identificar unívocamente la transacción.
2. B envía a A un mensaje encriptado con la clave pública de A, conteniendo el código temporario N_1, y un nuevo código temporario N_2. Dado que sólo B pudo desencriptar el mensaje original, la presencia de N_1 en el mensaje le asegura a A que está hablando con B.
3. A devuelve N_2, encriptado con la clave pública de B.
4. A elige un clave secreta y se la envía a B, encriptada primero con la clave pública de A (para asegurar que A la envió) y luego envia con la clave pública de B (de manera que sólo B la pueda leer).
5. B recupera la clave con la clave pública de A y su clave privada. De esta manera se asegura confidencialidad y autenticación en el intercambio de la clave secreta.

Análisis comparativo de los Sistemas de Administración de Claves

La comparación de los esquemas de manejo y distribución de claves que pueden recibir el nombre de convencionales (jerárquico y centralizado), con los esquemas de manejo y distribución de claves en un sistema asimétrico de clave pública, arroja una serie de resultados que pueden ser resumidos como sigue:

- En los sistemas convencionales la seguridad queda reducida a la protección de las claves maestras de más alto nivel. En caso que las claves estén seguras, el sistema permanecerá seguro, pero si se ven comprometidas, queda comprometido todo el sistema. En este sentido, la situación es idéntica en el sistema de gestión de clave pública que guarda las claves en algún medio de almacenamiento.

- Por otro lado, un sistema convencional en el que las claves son transmitidas, puede ser vulnerado por la escucha de la comunicación, mientras que en un sistema de gestión que use encriptado asimétrico eso no es posible.

- Sin embargo, en cualquier caso se necesita un agente de seguridad que haga una distribución inicial de las claves, con lo que la comparación de ambos sistemas no supone grandes diferencias en cuanto a aportes en seguridad.

Referencias

- [Everton,78] Everton J.K. " A hierarchical basis for encription key management in a computer communications network". Trends and applications 1978. Distributed processing, IEEE Computer Society. 1978.

- [IBM,78] IBM System Journal. Vol. 17-2. Monographic publication on cryptography. 1978.

- [KOHN,78] Kohnfelder, L. "Towards a practical public key cryptosystems". B.S. Thesis. M.I.T. Cambridge, Mass. 1978.

- [MERK, 78] Merkle, R and Hellman, M. "Hiding Information and Signatures in Trap Door Knapsacks" IEEE Transactions on Information Theory, September, 1978.

V. - NOVEDADES EN CRIPTOGRAFIA

VI. 1. PROYECTO CLIPPER

VI. 1.1 - Capstone

Capstone es el proyecto a largo plazo del gobierno de los Estados Unidos para desarrollar un conjunto de estándares para los sistemas criptográficos al alcance del público en general, como lo autoriza la Ley de Seguridad Informática de 1987 ("Computer Security Act"). Las agencias primordialmente responsables de Capstone son NIST (National Institute of Standards and Technology) y NSA (National Security Agency). El plan requiere que los elementos de Capstone se conviertan en estándares oficiales, en cuyo caso tanto el gobierno como las empresas privadas que hagan negocios con el gobierno tendrán que utilizar el sistema Capstone.

Existen cuatro componentes principales de Capstone: un algoritmo de encriptación masivo de datos, un algoritmo de firma digital, un protocolo de intercambio de claves y una función de "hash". El algoritmo de encriptación se denomina *Skipjack*, pero se lo conoce como *Clipper* que es el chip de encriptado que incluye al Skipjack. El algoritmo de firma digital es *DSS* y la función de "hash" es *SHS*. El protocolo de intercambio de claves no ha sido anunciado todavía.

Todas los componentes de Capstone disponen de una seguridad de 80 bits: todas las claves involucradas tienen 80 bits de longitud y sus otros aspectos se han diseñado para resistir un ataque de 80 bits, es decir, un esfuerzo de 2^{80} operaciones. Con el tiempo, el gobierno planea ubicar todo el sistema criptográfico Capstone en un solo chip.

VI. 1.2 - Clipper

Clipper es un chip de encriptado desarrollado y patrocinado por el gobierno de los Estados Unidos como parte del proyecto Capstone. Anunciado por la Casa Blanca en abril de 1993, Clipper fue diseñado para equilibrar los intereses de las fuerzas de seguridad con los intereses de los ciudadanos y de las industrias. Las fuerzas federales de seguridad que desean acceder a las comunicaciones de personas sospechadas de actividades ilícitas, por ejemplo, interceptando comunicaciones, se encuentran amenazadas por la seguridad de la criptografía. La industria y los ciudadanos, sin embargo, quieren comunicaciones seguras y buscan la solución en la criptografía.

La tecnología de Clipper apunta a satisfacer los intereses de ambas partes mediante una clave en depósito. La idea es que las comunicaciones deberían ser encriptadas con un algoritmo seguro pero las claves las guardarían una o más partes (entidades autorizadas para el depósito de claves) y serían accesibles para las fuerzas de seguridad cuando así lo autorice la Justicia. Por ejemplo, las comunicaciones personales serían inaccesibles a ojos no autorizados y las comunicaciones comerciales serían inaccesibles para el espionaje industrial. Sin embargo, el FBI podría escuchar las comunicaciones de terroristas o criminales.

Clipper fue propuesto como un estándar oficial de los Estados Unidos. Tendría que ser utilizado por quien realice negocios con el gobierno federal y para las comunicaciones internas del gobierno. Para cualquier otra persona, la utilización de Clipper es voluntaria. La empresa AT&T anunció un teléfono seguro que utiliza el chip Clipper.

El chip Clipper contiene un algoritmo de encriptado denominado Skipjack cuyos detalles no se han dado a publicidad. Cada chip contiene una unidad de clave U de 80 bits, que está en depósito dividida en dos partes en las entidades autorizadas para el depósito de claves. Ambas partes deben ser conocidas para poder recuperar la clave. También presenta un número de serie y una clave de familia, F, de 80 bits que es común a todos los chips Clipper. El chip se fabrica de modo que no pueda dilucidarse a partir del chip ni las claves ni el algoritmo Skipjack.

Cuando dos partes deciden comunicarse, primero acuerdan una "clave de sesión" K de 80 bits. El método por el cual eligen la clave queda librado al criterio del implementador; un método de clave pública como RSA o Diffie-Hellman son elecciones apropiadas. El mensaje se encripta con la clave K y se envía. Esta clave K no es una clave en depósito. Aparte del mensaje encriptado, otra parte de la información, llamada el LEAF (Campo de Acceso de las Fuerzas de Seguridad, "Law Enforcement Access Field") se crea y envía. Incluye la clave de sesión K encriptada con la unidad de clave U y luego se concatena con el número de serie del remitente y una cadena de autenticaciones y, por último, se encripta con una clave de familia. Los detalles exactos del campo de las fuerzas de seguridad permanecen secretos.

El receptor desencripta el LEAF, comprueba la cadena de autenticación y desencripta el mensaje con la clave K.

Supongamos que las fuerzas de seguridad desean intervenir una línea de comunicaciones. Utilizan la clave de familia para desencriptar el campo de acceso que le corresponde, las fuerzas de seguridad conocen el número de serie y poseen una versión encriptada de la clave de sesión. Presentan una orden autorizando a las dos entidades de depósito de claves, junto con el número de serie. Las entidades de depósito le proporcionan cada una las dos mitades de la clave y luego proceden al desencriptado para obtener la clave de sesión K. Las fuerzas de seguridad luego utilizan K para desencriptar el mensaje real.

Clipper es un proyecto que se encuentra bajo estudio. Tanto el poder Ejecutivo como el Legislativo de los EEUU lo están considerando y un panel de consulta recomendó un exhaustivo y extensivo debate público acerca de la política criptográfica. NIST ha invitado al público a enviar sus comentarios como parte de su propia revisión.

VI. 1.3 - Skipjack

Skipjack es el algoritmo de encriptado contenido en el chip Clipper que fue diseñado por la NSA. Utiliza una clave de 80 bits para encriptar bloques de datos de 64 bits. La misma

clave se utiliza para el desencriptado. Skipjack se puede utilizar del mismo modo que DES y puede ser más seguro que éste último ya que utiliza claves de 80 bits y trabaja la información en 32 pasos o "vueltas", mientras que DES utiliza claves de 56 bits y trabaja la información sólo 16 veces.

Los detalles de Skipjack son confidenciales. La decisión de no publicar los detalles del algoritmo fue criticada. Mucha gente sospecha que Skipjack no es seguro, ya sea debido a un descuido por parte de sus diseñadores o por la inclusión deliberada de una puerta-trampa secreta. Por otro lado, se ha tratado de descubrir las debilidades del DES a través de los años, ya que sus detalles son de conocimiento público. Estos numerosos intentos (y el hecho de que hayan fallado) han hecho que se considere al DES como confiable. Como Skipjack no es público, no se lo puede analizar y por ello el nivel de confianza debido puede no llegar a manifestarse.

Otra consecuencia de la confidencialidad de Skipjack consiste en que no puede implementarse en versión software sino solo en la versión hardware cuya fabricación ha sido autorizada por el gobierno.

VI. 1.4 - Digital Signature Standard (DSS)

DSS significa Estándar de Firma Digital ("Digital Signature Standard"), y requiere un DSA (Algoritmo de Firma Digital, "Digital Signature Algorithm"). Es parte del proyecto Capstone. Fue seleccionado por NIST en cooperación con la NSA para ser el estándar de autenticación digital del gobierno de los Estados Unidos. Se encuentra todavía en discusión si el gobierno debería adoptarlo como estándar, o no.

DSS se basa en el problema del logaritmo discreto y deriva de los sistemas criptográficos propuestos por Schnorr [SCHNO, 90] y ElGamal [ElGamal, 85]. Solo sirve para autenticación.

La crítica a DSS se concentra en los siguientes aspectos:

- no es posible el intercambio de claves

- el sistema criptográfico subyacente es demasiado reciente y no se lo ha estudiado lo suficiente como para que los usuarios confíen en su resistencia al ataque
- la verificación de las firmas con DSS es demasiado lenta
- la existencia de un segundo estándar de autenticación les causará muchos inconvenientes a los vendedores de software y hardware que ya han estandarizado al RSA
- el proceso por el cual NIST eligió a DSS fue secreto y arbitrario con la fuerte influencia de la NSA.

En el sistema DSS, la generación de una firma es más fácil que la verificación, mientras que en el sistema RSA, la verificación de la firma es más rápida que la generación de la misma (si los exponentes privados y públicos se eligen con esta propiedad, lo que es usualmente el caso). NIST reclama que este aspecto es una ventaja de DSS pero muchos involucrados con la criptografía piensan que es mejor que la verificación sea una operación rápida.

Las críticas más serias respecto a DSS apuntan a la seguridad. En un principio DSS se propuso con un tamaño fijo de clave de 512 bits. Después de mucha crítica en este aspecto, NIST efectuó una revisión de DSS para permitir el uso de claves de hasta 1024 bits. Más grave es el hecho de que DSS no ha estado en el mercado lo suficiente como para probar su resistencia ante el ataque y aunque el problema del logaritmo discreto es antiguo, la forma en que se lo utiliza en DSS fue propuesta primero por Schnorr en 1989 para su uso criptográfico y no ha recibido demasiado análisis público. En general, cualquier sistema criptográfico nuevo podría presentar serias fallas que solo se descubren después de varios años de estudio por parte de los criptógrafos.

Algunos investigadores advierten acerca de la existencia de números primos puerta-trampa en DSS, lo que facilitaría un ataque. Estos números son casos raros y se pueden evitar si se utilizan los procedimientos correctos en la generación de claves.

VI. 2. SECURE HYPERTEXT TRANSFER PROTOCOL (SHTTP)

VI. 2.1 - Introducción

La World Wide Web (WWW) es un sistema de hipermedia distribuido, el cual ha alcanzado una gran aceptación por los usuarios de Internet. Aunque los browsers de la WWW soportan protocolos de aplicación de Internet preexistentes, el protocolo nativo y primario usado entre los servidores y los clientes es el Hypertext Transfer Protocol (HTTP). El fácil uso del Web ha ido extendiendo el interés por su empleo como una arquitectura cliente/servidor para muchas aplicaciones. Muchas de tales aplicaciones requiere que el cliente y el servidor estén habilitados para la autenticación entre ellos y el intercambio de información confidencial. El HTTP original tuvo sólo un apoyo moderado por los mecanismos criptográficos apropiados para tales transacciones.

SHTTP (Secure Hypertext Transfer Protocol) fue diseñado por E. Rescorla y A.Schiffman para proveer conexiones HTTP seguras. SHTTP provee una amplia variedad de mecanismos para proveer confidencialidad, autenticación e integridad. La separación de mecanismos y políticas fue un objetivo explícito. El sistema no se limita a un sistema criptográfico particular, infraestructura de clave, o formato criptográfico.

SHTTP provee mecanismos de comunicaciones seguros entre un par de cliente-servidor HTTP, con el objeto de habilitar transacciones comerciales espontáneas en un gran rango de aplicaciones.

VI. 2.2 - Resumen de Características

SHTTP es un protocolo de comunicaciones seguro orientado a mensajes y diseñado para el uso conjunto con HTTP. Está diseñado para coexistir con el modelo de mensajes HTTP y

para integrarse fácilmente con las aplicaciones HTTP. Consecuentemente, imita mucho el estilo y la sintaxis del HTTP.

SHTTP provee una variedad de mecanismos de seguridad para servidores y clientes HTTP, otorgando opciones de servicios de seguridad apropiados para el amplio rango de posibles y potenciales usuarios finales de la WWW. El protocolo provee capacidades simétricas tanto para los clientes como para los servidores (ese trato igualitario está dado tanto para solicitudes como para respuestas, como para las preferencias de ambas partes) mientras que preservan el modelo de transacción y las características de implementación del HTTP.

Muchos formatos de mensajes criptográficos estándares pueden ser incorporados en servidores y clientes SHTTP, particularmente, pero en principio no limitado a [PKCS-7] y [MOSS]. SHTTP soporta interoperaciones a lo largo de una variedad de implementaciones, y es compatible con HTTP. En SHTTP los clientes conscientes pueden comunicarse con servidores inconscientes y viceversa, aunque obviamente tales transacciones no utilizarán las propiedades de seguridad de SHTTP.

SHTTP no requiere del lado del cliente certificados de clave pública (o claves públicas), dado que soporta modos simétricos de operación de clave única. Esto es importante porque significa que pueden ocurrir transacciones espontáneas privadas sin que usuarios individuales tengan que tener una clave pública establecida. Mientras SHTTP está habilitado para beneficiarse con las infraestructuras certificantes que se encuentran por todas partes, su utilización no requiere de éstas.

SHTTP soporta transacciones seguras punto a punto, en contraste con el original mecanismo de autorización de HTTP el cual requería que el cliente intentara el acceso y que fuera denegado antes de que el mecanismo de seguridad fuera empleado. Los clientes pueden ser priorizados para iniciar una transacción segura (típicamente usando información dada en

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

la cabecera del mensaje); esto puede ser usado, por ejemplo, para soportar encriptación de formularios. Con SHTTP, los datos no críticos necesitan ser enviados por la red en forma de texto plano.

SHTTP provee gran flexibilidad de algoritmos criptográficos, modos y parámetros. La opción de negociación es usada para permitir a los clientes y servidores ponerse de acuerdo en el modo de transacción (por ejemplo si la pregunta deberá estar firmada o encriptada o ambas cosas y similarmente para la respuesta); algoritmos criptográficos (RSA vs DSA para la firma digital, DES vs RC2 [RC2, Rivest] para la encriptación, etc.); y la selección de certificados (por favor firmar con su certificado de "Block-buster").

VI. 2.3 - Modos de Procesamiento

VI. 2.3.1. Preparación del Mensaje

La creación de un mensaje SHTTP puede ser pensado como una función con tres entradas:

1. El mensaje como texto plano (plaintext). Esto es un mensaje HTTP o algún otro objeto de datos.
2. Las preferencias criptográficas del receptor y la información de claves. Esto es explícitamente especificado por el receptor o sujeto a un conjunto de preferencias.
3. Las preferencias criptográficas del emisor y la información de claves. Esta entrada de la función puede ser pensada como implícita desde que existe sólo en la memoria del emisor.

Con el objeto de crear un mensaje SHTTP, entonces el emisor integra sus preferencias con las del receptor. El resultado de esto es una lista de especificaciones criptográficas para ser aplicadas e información de claves para ser usada en su aplicación. Esto puede requerir cierta intervención del usuario. Por ejemplo, podría haber múltiples claves disponibles para firmar el mensaje. Usando estos datos, el emisor aplica las especificaciones al mensaje de texto plano para crear el mensaje SHTTP.

VI. 2.3.2 - Recuperación del Mensaje

La recuperación de un mensaje SHTTP puede ser pensada como una función de cuatro entradas distintas:

1. El mensaje SHTTP
2. Las preferencias criptográficas establecidas del receptor y la información de claves. El receptor tiene la oportunidad de recordar que preferencias criptográficas proveyó de manera que sean desreferenciadas para este documento.
3. Las preferencias criptográficas actuales del receptor y la información de claves.
4. Las preferencias criptográficas previamente establecidas por el emisor. El emisor puede haber establecido que ejecutaría ciertas operaciones criptográficas sobre ese mensaje.

Con el objeto de recuperar un mensaje SHTTP, el receptor necesita leer la cabecera (header) para descubrir cuales transformaciones criptográficas fueron ejecutadas en el mensaje, luego remover las transformaciones utilizando alguna combinación de la información de claves del receptor y del emisor, mientras se toma nota de que operaciones se han aplicado.

El receptor puede elegir verificar que las operaciones aplicadas coincidan tanto con las que el emisor dijo que realizaría (entrada 4) como con las que el receptor requirió (entrada 2) como también las preferencias actuales para ver si el mensaje SHTTP fue transformado correctamente.

VI. 2.4 - Modos de Operación

La protección del mensaje puede ser provista desde tres ángulos distintos: firma, autenticación y encriptación. Cualquier mensaje puede ser firmado, autentificado y encriptado o cualquier combinación de los tres (incluyendo ninguna protección).

Se soportan mecanismos de manejo de claves múltiples, incluyendo passwords compartidos manualmente, intercambio de claves por claves públicas y distribución de tickets Kerberos [KERBEROS, 88]. En particular, la provisión de claves se suele realizar en una sesión simétrica preacordada (en una transacción previa, o fuera de banda) de manera de enviar mensajes confidenciales quienes no tienen claves públicas.

Además, se provee de un mecanismo de respuesta que permita asegurar a las partes la actualidad de las transacciones.

VI. 2.4.1 Firmas

Si se aplica la firma digital, se puede agregar un certificado al mensaje, o bien esperar que el receptor obtenga el certificado requerido independientemente.

VI. 2.4.2 Intercambio de claves y encriptación

SHTTP define dos mecanismos de transferencia de claves, uno usando intercambio de claves por clave pública y el otro claves externamente arregladas.

En el primer caso, la clave simétrica es enviada encriptada con la clave pública del receptor. En el segundo, se encripta el contenido utilizando una clave de sesión preestablecida, con la información de identificación de clave especificada en una de las líneas del encabezado. Las claves pueden ser también tomadas de tickets Kerberos.

VI. 2.4.3 - Integridad del mensaje y Autenticación del emisor

SHTTP provee de un medio para verificar la integridad del mensaje y la autenticidad del **emisor** de un mensaje por medio del cómputo de un Código de Autenticación de Mensaje (CAM). Esta técnica no requiere el uso de criptografía por clave pública o de encriptación.

Este mecanismo es también útil cuando es apropiado permitir a las partes identificarse mutuamente de una manera confiable en una transacción. La provisión de este mecanismo está motivada por nuestro prejuicio de que la acción de "firmar" una transacción debe ser explícita y consciente para el usuario, mientras que muchas necesidades de autenticación

(como por ejemplo el control de acceso) pueden alcanzarse con un mecanismo simple que mantenga las ventajas de la criptografía de clave pública para el intercambio de claves.

VI. 2.4.4 - Actualidad

El protocolo provee un mecanismo de respuesta simple, el que asegura a ambas partes la actualidad de las transmisiones. Además, la protección de integridad provista a los encabezados HTTP permite implementaciones que consideren al encabezado Date (fecha), permitido en mensajes HTTP, como un indicador de la actualidad del mensaje, en los casos que sea apropiado.

VI. 2.4.5 - Opciones de Implementación

De manera de animar la adopción masiva de transmisiones seguras en la WWW facilitadas por un pequeño rango de requerimientos de aplicaciones, variabilidad en la sofisticación del usuario y condiciones de implementación dispares, SHTTP ofrece deliberadamente una gran variedad de opciones de implementación, cuya discusión y análisis no son de interés en el presente trabajo.

VI. 2.4.6 - Formato del mensaje

La sintaxis del SHTTP imita la del HTTP en un esfuerzo de fácil integración con sistemas que ya procesan HTTP. Además, ciertos encabezados HTTP son utilizados como encabezados SHTTP porque proveen funcionalidad que tiene implicaciones de seguridad.
Un mensaje SHTTP consiste en una línea de pedido o estado (como en HTTP) seguida por una serie de encabezados tipo RFC-822 seguidos por contenido encapsulado. Una vez que el contenido ha sido recuperado, éste será otro mensaje SHTTP, un mensaje HTTP o simplemente información.

Para la compatibilidad de implementaciones HTTP existentes, se distinguen los pedidos y respuestas de transacciones SHTTP con un indicador de protocolo distinto ("Secure-HTTP/1.2").

El formato de la **línea de pedido** del SHTTP es similar a la del HTTP. Sin embargo, todos los pedidos de SHTTP usan el método "Secure" (Seguro). Todos los pedidos de SHTTP deberán decir (si se utiliza esta versión del protocolo):

Secure * Secure-HTTP/1.2

Se aceptan variaciones de mayúsculas y minúsculas. El asterisco es un separador y será ignorado por los servidores.

La primera **línea de la respuesta** del servidor deberá decir:

Secure-HTTP/1.2 200 OK

haya el pedido tenido éxito o no. Esto evita el análisis de éxito o no de cualquier pedido, lo que el receptor correcto puede determinar de la información encapsulada. Nuevamente, se aceptan variaciones de mayúsculas y minúsculas.

El encabezado de cada mensaje SHTTP puede contener hasta siete líneas de las cuales cinco son opcionales y no entraremos en detalles sobre éstas en este trabajo.

VI. 2.5 - Debilidades

El uso de una banda de intercambio de claves es potencialmente muy problemático, los autores no dedican demasiado tiempo en asegurar que las claves sean transferidas apropiadamente. Una transferencia inapropiada podría ser un esquema tal que envía la clave B como Ea(B). Esto quiere decir, que la clave B que reemplaza a la clave A no puede ser enviada usando la clave A para encriptarla. Si un atacante ha interceptado la clave A, entonces puede tener la clave B, y el cambio de clave es una pérdida de tiempo (con respecto a este ataque). Exactamente este error fue cometido por los Japoneses en la Segunda Guerra Mundial. No es conveniente esperar que los programadores aprendan de los errores de otros, especialmente de los errores que persisten desde hace 50 años.

Por ser tan flexible el SHTTP puede resultar un arma de doble filo para el programador. Cierto es que esto no ofrece muchas opciones para quebrantarlo, pero parece no tener una actitud como "encripte todo sin esfuerzo".

Un programador, especialmente uno que no esté familiarizado con los problemas de la seguridad y la criptografía, podría pensar que "el uso del SHTTP me protegerá", al pensar ésto puede estar cometiendo un error, al no proporcionar ninguna protección criptográfica a la información. La probabilidad de que ésto ocurra no es muy alta, pero vale la pena considerar el problema.

Referencias

- [ElGamal, 85] Un criptosistema de Clave Pública y un esquema de firma basado en logaritmo discreto. IEEE *Transactions on Information Theory*, 1985.

- [KERBEROS, 88] Sistema de autenticación en red con clave secreta que utiliza el sistema DES para encriptado y autenticación para el acceso a los recursos de red. Fue desarrollado en MIT (J. G. Steiner, B. C. Neuman and J. I. Schiller), Febrero 1988.

- [PKCS] Estándares de criptografía con Clave Pública (Public Key Cryptography Standards). Consiste en un conjunto de estándares diseñados por RSA Data Security Inc.

- [RC2, Rivest] Algoritmo de encriptado de bloque simétrico con tamaño de clave variable. Es aproximadamente el doble de rápido que el DES.

- [SCHNO, 90] .Schnorr, C. P. Efficient identification and signatures for smart cards. New York, 1990.

VII. – SISTEMA *Módulos para la Transferencia Segura y Secreta de Información*

El sistema permite la transferencia de información en forma segura y secreta entre procesos, lo cual es transparente para el usuario.

VII. 1 – Descripción

Como mecanismo de seguridad se implementaron distintos tipos de algoritmos de encriptación de datos. Los algoritmos utilizados para la transmisión de mensajes son el CAESAR (Método utilizado por Julio César), DES (Data Encription Standard) y el RSA (Rivest Shamir Adleman). Asimismo, se consideró un cuarto método de encriptación de mensajes, que llamamos RSA CERTIFIED, que es una variante del algoritmo RSA con la inclusión del uso de certificados de claves públicas.

Por otro lado, para la transmisión de mensajes se utilizaron como archivos de trabajo los archivos de texto, que en el ambiente de transferencia se identifican como "mensajes".

El sistema consta de las siguientes partes:

- Usuarios
- Editores
- Mensajes
- Administrador

Los ***Usuarios*** del sistema se caracterizan con una identificación (nombre del usuario), una clave personal (password) y un directorio de trabajo, desde y donde van a recibir y enviar mensajes con otros usuarios del sistema. Para este sistema se ha definido que los directorios de trabajo se establezcan todos dentro de la misma unidad de disco.

Se definieron tres tipos de editores para la visualización y edición de los archivos involucrados en la transferencia de información. Estos editores son:

- ***Editor de Texto***

  Permite la edición y visualización de los archivos de texto (*.txt) para encriptar y enviar, llamados en el ambiente criptográfico *plaintext* y los archivos desencriptados y recibidos (*.ame).

- ***Editor Criptográfico***

  Permite la visualización y edición de los archivos encriptados con los diferentes métodos criptográficos que forman parte de la transmisión de información. Se pueden ver los archivos encriptados bajo el método CAESAR (*.cae), el DES (*.des), el RSA (*.rsa) y el RSA CERTIFIED (*.rsc).

- ***Visor de Certificados***

  Permite la visualización de certificados de clave pública, que forman parte de los procesos de encriptación y desencriptación haciendo uso del método RSA CERTIFIED. Se pueden visualizar los certificados creados por la Autoridad Certificante (*.cfa) ante un pedido de un Usuario del Sistema o el certificado obtenido por el receptor al recibir el mensaje y desencriptarlo (*.cfo), y el cual le permite verificar la autenticidad del emisor.

Los usuarios del sistema pueden efectuar el ***envío de mensajes*** y la ***recepción de mensajes*** en forma segura y secreta manteniéndose esta transmisión transparente para los usuarios involucrados.

Para enviar mensajes, primero el usuario emisor debe identificarse y seleccionar el usuario a quien le enviará el mensaje (usuario receptor). Luego se deberá seleccionar el método de encriptación a utilizar para la transmisión y finalmente el texto que se desea enviar. Obviamente el usuario emisor y receptor no pueden ser los mismos.

Para la recepción de mensajes, el usuario se debe identificar, seleccionar el método de encriptación a utilizar para desencriptar el mensaje y el texto recibido.

Corresponde señalar que tanto para el envío como la recepción, los mensajes (archivos de texto) se encuentran en los directorios definidos como de trabajo. Un usuario siempre recibirá sus mensajes en su directorio de trabajo y siempre enviara desde su directorio de trabajo.

Se ha definido para la administración y distribución de las claves públicas y privadas un ***Administrador***, donde se pueden generar las claves públicas y privadas de cada usuario, incluso se pueden revocar claves. Asimismo, el administrador permite la visualización del Directorio público de claves y los Certificados otorgados por la Autoridad Certificante a pedido de los distintos usuarios del Sistema.

VII. 2 – Método RSA CERTIFIED

El método es una combinación del algoritmo de encriptación RSA con el uso de certificados de claves públicas que permiten la encriptación de mensajes haciendo uso de un sistema criptográfico asimétrico.

El método RSA CERTIFIED se compone de los siguientes pasos :

1. Se solicita a la Autoridad Certificante un certificado de clave pública, que autentique la validez de la clave pública del receptor.
2. La Autoridad Certificante emite y envia al solicitante (usuario enviador) un Certificado que incluye el nombre del solicitante, la clave pública del usuario receptor, fecha y hora de emisión y un código de sesión que se genera en ese momento.
3. La Autoridad certificante encripta el código de sesión (archivo *.cfa) generado con la creación del certificado utilizando la clave privada de la AC (archivo *.cfe).
4. Utilizando la clave pública del usuario receptor se encripta el mensaje a enviar (*.txt) y el certificado generado por la AC (*.cfa), obteniéndose bajo el método RSA el archivo encriptado (*.rsc).
5. El usuario receptor recibe el mensaje y lo desencripta utilizando su clave privada, obteniéndose los archivos descifrado (*.ame) y certificado (*.cfd).

6. Se desencripta el certificado con la clave pública de la AC obteniéndose el archivo desencriptado (*.cfo).

La generación y encriptación de los certificados a través del uso de este algoritmo resulta transparente para los usuarios involucrados.

VII. 3 - Proyección del Trabajo

El propósito del trabajo fue generar una herramienta de prueba y a su vez permitir el desarrollo de trabajos de investigación en el futuro, que utilicen este ambiente como base para el estudio de un concepto particular de la criptografía y/o del criptoanálisis.

Existen diversos elementos que no fueron incluidos en el desarrollo del presente trabajo, dado que no contribuían en esencia al objetivo planteado. Por ello es que no se implementaron como parte del sistema, pero la incorporación de estos elementos podrían otorgarle más potencialidad y generalidad como ambiente de pruebas criptográficas. A continuación exponemos algunas de las alternativas que se podrían incluir en el sistema y que planteamos como proyección del trabajo.

Además de los métodos criptográficos implementados en el trabajo (CAESAR, DES y RSA), existen otros algoritmos que responden a los métodos de sustitución y transformación que podrían ser incluidos. En este orden, se utilizan para la encriptación de mensajes combinaciones de los algoritmos anteriormente mencionados como así también se utilizan variantes de los algoritmos simétricos y asimétricos tal como el triple DES entre otros, lo cuales podrían ser tenidos en cuenta para incluir como métodos de encriptación en el sistema.

Por otro lado, el sistema permite la encriptación de archivos de texto, pero se podrían implementar estos algoritmos generalizándolos para la encriptación de cualquier tipo de archivos, como ser archivos de bases de datos, archivos comprimidos, archivos de imágenes y

otros. Si bien el trabajo no se desarrolló para la encriptación de cualquier tipo de archivos, los algoritmos DES y RSA internamente fueron desarrollados para trabajar a nivel de bits, por lo cual resultaría fácilmente adaptable.

Otro de los aspectos a considerar es la implementación del sistema no centralizado en donde cada usuario puede trabajar con su disco de trabajo. El sistema podría mantener un disco compartido para el almacenamiento del directorio de claves públicas y donde cada usuario remoto defina su propio directorio en cualquier unidad de disco conocida que forme parte de la red virtual o físicamente definida. Bajo este esquema se generalizaría la implementación del sistema en cualquier red distribuida.

VIII. – CONCLUSIONES

En la actualidad, dado el avance tecnológico informático y en las telecomunicaciones, la principal preocupación es cómo dar seguridad a la información que es transmitida a través de las redes de comunicación, en bancos de datos, en cajeros automáticos, etc. Es por ello nuestro interés de estudiar los métodos criptográficos que se utilizan como mecanismos para dar seguridad.

Una realidad muy conocida es que la mayoría de los usuarios prefieren para la elección de sus claves secretas (passwords, pines, etc.), nombres, fechas de nacimiento y otros datos de personas queridas. Esto resulta razonable, desde que un usuario elige los códigos que más se acuerda, pero no hay métodos criptográficos que resuelvan esta debilidad y vulnerabilidad de la información que se tiene a partir de la elección de códigos lógicos y fácilmente deducibles. Un oponente puede intentar encontrar la clave utilizando palabras de un diccionario, nombres en libros de nombres propios, y otras alternativas de pruebas exhaustivas.

Encriptar significa tornar un mensaje legible (el texto plano) en uno ilegible (el texto encriptado) de acuerdo con una fórmula matemática. Obviamente, encriptar información tiene sentido únicamente si es posible desencriptarla, es decir volver a recuperar el texto plano original en base al texto encriptado. Este proceso de encriptar y desencriptar carecería de utilidad si fuera igual de fácil para todos desencriptar la información que se quiere mantener o transmitir en forma secreta. En realidad es extremadamente difícil desencriptar la información, pudiendo tomar hasta millones de años de análisis por las computadoras más poderosas. Esto es, claro, a menos que se conozca la "clave".

La ventaja principal del uso de claves públicas es su mayor seguridad. Las claves privadas no se transmiten ni se revelan a persona alguna. En el sistema con clave secreta, siempre existe la posibilidad de que sea descubierta durante la transmisión.

Los sistemas con clave pública proporcionan asimismo un método de firma digital. La autenticación por medio de sistemas con clave secreta requiere que algunos secretos se compartan y algunas veces hasta requiere de la confianza en un tercero. El remitente puede alegar que desconoce un mensaje que previamente ha firmado diciendo que la clave compartida no fue mantenida en el debido secreto por alguna de las partes. La autenticación con clave pública evita este tipo de inconvenientes ya que cada usuario es responsable de mantener en secreto su clave privada. Por ello, la autenticación con clave pública no es objetable.

La desventaja de utilizar el método de encriptación con clave pública es la velocidad: existen métodos de encriptación con clave privada que funcionan a mayor velocidad que cualquier método de encriptación con clave pública pero ambos métodos pueden combinarse para obtener lo mejor de cada uno: la ***seguridad*** de la clave pública y la ***velocidad*** de la clave secreta.

El RSA es el esquema por excelencia de claves públicas. Este agrega dos funciones importantes que no proporciona el DES (que es el esquema por excelencia de claves secretas): el intercambio seguro de claves sin un previo intercambio de claves secretas y la firma digital.

El método DES o cualquier otro sistema de encriptación, tal como los métodos de sustitución (CAESAR), transposición o combinaciones de estos últimos, son preferibles de utilizar para encriptar mensajes largos, ya que resultan más veloces que el RSA.

En algunas situaciones, el RSA no es necesario y basta con el DES. Esto incluye a los entornos con multiusuarios donde es posible intercambiar claves DES en forma segura, por ejemplo, si ambas partes se reúnen en privado.

El sistema RSA es poco necesario en un entorno con un solo usuario. Por ejemplo, si se desea mantener encriptados los archivos personales, solo se necesita el DES o cualquier otro sistema de encriptación como los estudiados en este trabajo. El RSA y la criptografía con clave pública, en general, se aplican mejor en entornos con multiusuarios. Cualquier sistema que utilice firmas digitales necesita del RSA o de algún otro sistema con clave pública.

Finalmente, no existe un algoritmo universalmente aplicable para todas los sistemas de información. Se deberá seleccionar el método criptográfico que resulte más apropiado y efectivo, para obtener el nivel de seguridad y confidencialidad de la información que se desea que el sistema informático alcance.

ANEXO

/* Biblioteca de rutinas de encriptación y desencriptación Algoritmo Julio Cesar */

```
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <windows.h>

void far pascal _export EncCaesar(char *fPlano, char *fCrypt, int desp)
{
 FILE *file1, *file2;
 int c,resul;

 file1 = fopen(fPlano,"rb");
 if (file1==NULL)
    {
    MessageBox(NULL,"No encontró archivo origen","Error",0);
    printf("Error , NO se encontró el archivo %s !!!!\n",fPlano);
    }
 else
    {
     file2 = fopen(fCrypt,"wb");
     if (file2==NULL)
        MessageBox(NULL,"No encontró el archivo destino","Error",0);
     while((c=fgetc(file1))!=EOF)
       {
        resul=fmod(c+desp,255);
        fputc(resul,file2);
       }
     fclose(file1); fclose(file2);
    }
  return;
}

void far pascal _export DesCaesar(char *fCrypt,char *fPlano,int desp)
{
 FILE *file1, *file2;
 int c,resul;

 if ((file1=fopen(fCrypt,"rb"))==NULL)
    MessageBox(NULL,"No se encontró el Archivo","Error",0);
 else
    {
     file2=fopen(fPlano,"wb");
     while((c=fgetc(file1))!=EOF)
       {
        resul=fmod(c+255-desp,255);
        fputc(resul,file2);
       }
     fclose(file1); fclose(file2);
    }
 return;
}
```

/* Biblioteca de rutinas para encriptación y desencriptación con DES */

```
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <string.h>
#include <stdlib.h>
#include <windows.h>
#include "deslib.h"

static void rotate (unsigned char *c, int n);
static long S (ks_type ip);
static int fourbits (ks_type k, int s);
static int sixbits (ks_type k, int s);


void far pascal _export encriDES (char *argv,char *archivo,char *textDES)
{
 int i;
 struct LR op,ip;
 struct ks keys[16];

 FILE *forigen, *fe;

 if ((forigen=fopen(archivo,"rb"))==NULL)
   MessageBox(NULL,"No encontró el Archivo","Error",0);
 else
   fe = fopen(textDES,"wb");

 for (i=0;i<16;i++)
  keys[i] = KS(i,argv);

 while (fread(&ip,1,sizeof(struct LR),forigen) != 0)
  {
   int n;

  /* Permutación Inicial */
   permute(&op.L,&ip.L,(long *)IPtbl,64);

  /* Iteración de las Claves */
   for (n=0;n<16;n++)
    {
     ip.L = op.R;
     ip.R = op.L ^ f(op.R,keys[n]);
     op.R = ip.R;
     op.L = ip.L;
    }

  /* Permutación Final */
   inverse_permute(&op.L,&ip.L,(long *)IPtbl,64);
   fwrite(&op,1,sizeof(struct LR),fe);

  /* Completa el último bloque con ceros */
   ip.L = ip.R = 0;
  }
fclose(forigen);
```

```
 fclose(fe);
}


void far pascal _export decryptDES (char *argv,char *textoenc,char *Plano)
{
 int i;
 struct LR op,ip;
 struct ks keys[16];
 FILE *forigen, *fe;

 if ((forigen=fopen(textoenc,"rb"))==NULL)
   MessageBox(NULL,"No encontró el Archivo","Error",0);
 else
   fe = fopen(Plano,"wb");

 for (i=0;i<16;i++)
  keys[i] = KS(i,argv);
 while (fread(&ip,1,sizeof(struct LR),forigen) != 0)
  {
   int n;

  /* Permutación Inicial */
   permute(&op.L,&ip.L,(long *)IPtbl,64);

  /* Iteración de las Claves */
   for (n=15;n>=0;n--)
    {
     ip.R = op.L;
     ip.L = op.R ^ f(op.L,keys[n]);
     op.R = ip.R;
     op.L = ip.L;
    }

   /* Permutación Final */
   inverse_permute(&op.L,&ip.L,(long *)IPtbl,64);
   fwrite(&op,1,sizeof(struct LR),fe);

  /* Completa el último bloque con ceros */
   ip.L = ip.R = 0;
  }
 fclose(forigen);
 fclose(fe);
}


/* Permutación inversa de un string de 64 bits */
void inverse_permute(long *op, long *ip, long *tbl, int n)
{
 int i;
 long *pt = (long *) Pmask;

 *op = *(op+1) = 0;
 for (i=0;i<n;i++)
  {
  if ((*ip & *pt ) || (*(ip+1) & *(pt+1)))
    {
     *op |= *tbl;
     *(op+1) |= *(tbl+1);
```

```
      }
    tbl +=2;
    pt +=2;
    }
  }


  /* Permutación de un string de 64 bits */
  void permute(long *op, long *ip, long *tbl, int n)
  {
   int i;
   long *pt = (long *) Pmask;

   *op = *(op+1) = 0;
   for (i=0;i<n;i++)
    {
    if ((*ip & *tbl ) || (*(ip+1) & *(tbl+1)))
      {
       *op |= *pt;
       *(op+1) |= *(pt+1);
      }
    tbl +=2;
    pt +=2;
    }
  }


  /* Función f(R,K) */
  long f(long blk,ks_type key)
  {
   lr_type ir;
   lr_type or;
   union_type tr, kr;
   /* union {
       struct LR f;
       struct ks kn;
     } tr={0,0} , kr={0,0};
   */
   ir.L = 0;
   ir.R = 0;
   tr.f.L = 0;
   tr.f.R = 0;
   tr.kn.ki[0] = '\0';
   ir.L = blk;
   kr.kn = key;
   permute(&or.L, &ir.L,(long *)Etbl,48);
   tr.f.L ^= kr.f.L;
   tr.f.R ^= kr.f.R;
   ir.L = S(tr.kn);
   permute(&or.L, &ir.L,(long *)Ptbl,32);
   return or.L;
  }


  /* Se convierten los 48 bits de clave a 32 bits */
  static long S(ks_type ip)
  {
   int i;
   long op = 0;
```

```
  for (i=8;i>0;--i)
     {
      long four = fourbits(ip,i);
      op |= four << ((i-1)*4);
     }
    return op;
}


/* Extracción de 4 bits de la Clave */
static int fourbits (ks_type k,int s)
{
 int i = sixbits(k,s);
 int row,col;

 row = ((i >> 4) & 2) | (i & 1);
 col = (i >> 1) & 0xf;
 return stbl[8-s][row][col];
}


/* Extracción de 6 bits de pos s del bloque de claves */
static int sixbits(ks_type k, int s)
{
 int op = 0;
 int n = (8 - s);
 int i;

 for (i=0;i<2;i++)
   {
    int off = ex6[n][i][0];
    unsigned char c = k.ki[off];
    c >>= ex6[n][i][1];
    c <<= ex6[n][i][2];
    c &= ex6[n][i][3];
    op |= c;
   }
 return op;
}


/* Función KS */
ks_type KS(int n, char *key)
{
 static unsigned char cd[8];
 static int its[] = {1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1};
 union
   {
    struct ks kn;
    struct LR filler;
   } result;
 if (n == 0)
  permute((long *)cd, (long *)key, (long *)PC1tbl, 64);
 rotate(cd,its[n]);
 rotate(cd+4,its[n]);
 permute(&result.filler.L,(long *)cd,(long *)PC2tbl, 48);
 return result.kn;
}
```

```
/* Rotación de 4 bits de n posiciones a la izquierda */
static void rotate (unsigned char *c, int n)
{
  int i;
  unsigned j,k;

  k = *c >> (8 - n);
  for (i=3;i>=0;--i)
   {
    j = (*(c+i) << n) + k;
    k = j >> 8;
    *(c+i) = (unsigned char) j;
   }
}


/* Máscara de Permutaciones */
unsigned char Pmask[]={
 P( 1),P( 2),P( 3),P( 4),P( 5),P( 6),P( 7),P( 8),
 P( 9),P(10),P(11),P(12),P(13),P(14),P(15),P(16),
 P(17),P(18),P(19),P(20),P(21),P(22),P(23),P(24),
 P(25),P(26),P(27),P(28),P(29),P(30),P(31),P(32),
 P(33),P(34),P(35),P(36),P(37),P(38),P(39),P(40),
 P(41),P(42),P(43),P(44),P(45),P(46),P(47),P(48),
 P(49),P(50),P(51),P(52),P(53),P(54),P(55),P(56),
 P(57),P(58),P(59),P(60),P(61),P(62),P(63),P(64)
};


/* Tabla Inicial y Final de Permutación  */
unsigned char IPtbl[]={
 P(58),P(50),P(42),P(34),P(26),P(18),P(10),P( 2),
 P(60),P(52),P(44),P(36),P(28),P(20),P(12),P( 4),
 P(62),P(54),P(46),P(38),P(30),P(22),P(14),P( 6),
 P(64),P(56),P(48),P(40),P(32),P(24),P(16),P( 8),
 P(57),P(49),P(41),P(33),P(25),P(17),P( 9),P( 1),
 P(59),P(51),P(43),P(35),P(27),P(19),P(11),P( 3),
 P(61),P(53),P(45),P(37),P(29),P(21),P(13),P( 5),
 P(63),P(55),P(47),P(39),P(31),P(23),P(15),P( 7)
};

/* Tabla E de Permutación para la función f */
unsigned char Etbl[]={
 P(32),P( 1),P( 2),P( 3),P( 4),P( 5),
 P( 4),P( 5),P( 6),P( 7),P( 8),P( 9),
 P( 8),P( 9),P(10),P(11),P(12),P(13),
 P(12),P(13),P(14),P(15),P(16),P(17),
 P(16),P(17),P(18),P(19),P(20),P(21),
 P(20),P(21),P(22),P(23),P(24),P(25),
 P(24),P(25),P(26),P(27),P(28),P(29),
 P(28),P(29),P(30),P(31),P(32),P( 1)
 };


/* Tabla de Permutación P para la función f  */
unsigned char Ptbl[]={
 P(16),P( 7),P(20),P(21),P(29),P(12),P(28),P(17),
 P( 1),P(15),P(23),P(26),P( 5),P(18),P(31),P(10),
```

```
 P( 2),P( 8),P(24),P(14),P(32),P(27),P( 3),P( 9),
 P(19),P(13),P(30),P( 6),P(22),P(11),P( 4),P(25)};

/* Tabla de conversión de 6 bits en 4 bits  */
unsigned char stbl[8][4][16] = {
 /*------------------s1------------------*/
 14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7,
 0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8,
 4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0,
 15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13,
 /*------------------s2------------------*/
 15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10,
 3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5,
 0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15,
 13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9,
 /*------------------s3------------------*/
 10,0,9,14,6,3,15,5,1,13,12,7,11,4,2,8,
 13,7,0,9,3,4,6,10,2,8,5,14,12,11,15,1,
 13,6,4,9,8,15,3,0,11,1,2,12,5,10,14,7,
 1,10,13,0,6,9,8,7,4,15,14,3,11,5,2,12,
 /*------------------s4------------------*/
 7,13,14,3,0,6,9,10,1,2,8,5,11,12,4,15,
 13,8,11,5,6,15,0,3,4,7,2,12,1,10,14,9,
 10,6,9,0,12,11,7,13,15,1,3,14,5,2,8,4,
 3,15,0,6,10,1,13,8,9,4,5,11,12,7,2,14,
 /*------------------s5------------------*/
 2,12,4,1,7,10,11,6,8,5,3,15,13,0,14,9,
 14,11,2,12,4,7,13,1,5,0,15,10,3,9,8,6,
 4,2,1,11,10,13,7,8,15,9,12,5,6,3,0,14,
 11,8,12,7,1,14,2,13,6,15,0,9,10,4,5,3,
 /*------------------s6------------------*/
 12,1,10,15,9,2,6,8,0,13,3,4,14,7,5,11,
 10,15,4,2,7,12,9,5,6,1,13,14,0,11,3,8,
 9,14,15,5,2,8,12,3,7,0,4,10,1,13,11,6,
 4,3,2,12,9,5,15,10,11,14,1,7,6,0,8,13,
 /*------------------s7------------------*/
 4,11,2,14,15,0,8,13,3,12,9,7,5,10,6,1,
 13,0,11,7,4,9,1,10,14,3,5,12,2,15,8,6,
 1,4,11,13,12,3,7,14,10,15,6,8,0,5,9,2,
 6,11,13,8,1,4,10,7,9,5,0,15,14,2,3,12,
 /*------------------s8------------------*/
 13,2,8,4,6,15,11,1,10,9,3,14,5,0,12,7,
 1,15,13,8,10,3,7,4,12,5,6,11,0,14,9,2,
 7,11,4,1,9,12,14,2,0,6,10,13,15,3,5,8,
 2,1,14,7,4,10,8,13,15,12,9,0,3,5,6,11
 };


/* Selección 1 para el cálculo de la Clave PC-1 */
unsigned char PC1tbl[]={
 P(57),P(49),P(41),P(33),P(25),P(17),P( 9),P( 0),
 P( 1),P(58),P(50),P(42),P(34),P(26),P(18),P( 0),
 P(10),P( 2),P(59),P(51),P(43),P(35),P(27),P( 0),
 P(19),P(11),P( 3),P(60),P(52),P(44),P(36),P( 0),
 P(63),P(55),P(47),P(39),P(31),P(23),P(15),P( 0),
 P( 7),P(62),P(54),P(46),P(38),P(30),P(22),P( 0),
 P(14),P( 6),P(61),P(53),P(45),P(37),P(29),P( 0),
 P(21),P(13),P( 5),P(28),P(20),P(12),P( 4),P( 0)
};
```

```
/*   PC - 2    */
unsigned char PC2tbl[]={
 P(14),P(17),P(11),P(24),P( 1),P( 5),P( 3),P(28),
 P(15),P( 6),P(21),P(10),P(23),P(19),P(12),P( 4),
 P(26),P( 8),P(16),P( 7),P(27),P(20),P(13),P( 2),
 P(41),P(52),P(31),P(37),P(47),P(55),P(30),P(40),
 P(51),P(45),P(33),P(48),P(44),P(49),P(30),P(56),
 P(34),P(53),P(46),P(42),P(50),P(36),P(29),P(32)
};


/* Extracción de 6 bits de un string de 64 */
unsigned char ex6[8][2][4]={ /* byte , >> , << , & */
 0,2,0,0x3f,    /* s = 8 */
 0,2,1,0x3f,
/* s = 7 */
 0,0,4,0x30,
 1,4,0,0x0f,
 /* s = 6 */
 1,0,2,0x3c,
 2,6,0,0x03,
 /* s = 5 */
 2,0,0,0x3f,
 2,0,0,0x3f,
 /* s = 4 */
 3,2,0,0x3f,
 3,2,0,0x3f,
 /* s = 3 */
 3,0,4,0x30,
 4,4,0,0x0f,
 /* s = 2 */
 4,0,2,0x3c,
 5,6,0,0x03,
 /* s = 1 */
 5,0,0,0x3f,
 5,0,0,0x3f
};
```

/* Biblioteca que contiene las funciones de encriptación y desencriptación con RSA */

```
#include <math.h>
#include <stdio.h>
#include <windows.h>

void expmod (double m,double y,double n,double *res);


void far pascal _export cifraRSA (char *origen,char *destino,double
exp,double n)
/* cada caracter de origen ^exp mod n */
{
double res,m;
int f, entero;
FILE *fo, *fd;


if ((fo=fopen(origen,"rb"))==NULL)
     printf("Error , NO se encontró el archivo %s !!!!\n",origen);
else
     fd=fopen(destino,"wb");

while((f=fgetc(fo))!=EOF)
   {
    m = f;
    expmod(m,exp,n,&res);       /* Encripta exp = c */

    entero = res;
    fwrite(&entero,sizeof(entero),1,fd);
   }
f = 0;
fwrite(&f,sizeof(f),1,fd);
fclose(fo);
fclose(fd);
}


void far pascal _export descifraRSA (char *origen,char *destino,double
exp,double n)
/* cada caracter de origen ^exp mod n */
{
double res,m;
int entero;
FILE *fo, *fd;


if ((fo=fopen(origen,"rb"))==NULL)
     printf("Error , NO se encontró el archivo %s !!!!\n",origen);
else
     fd=fopen(destino,"wb");

fread(&entero,sizeof(entero),1,fo);
while (entero!= 0)
   {
```

```
    m = entero;
    expmod(m,exp,n,&res);       /* Desencripta exp = d */

    fputc(res,fd);
    fread(&entero,sizeof(entero),1,fo);
   }
fclose(fo);
fclose(fd);
}


void expmod (double m,double y,double n,double *res)
/* determina m^y mod n */
{
 double z,r,zl,r1,i;
 double x,uno;

 r = 1;
 x = y;
 z = fmod(m,n);  /* z = m mod n */

 while (x != 0)
   {
    i = fmod(x,2);
    if (i != 0)      /* Si y es impar */
      {
      r1 = r * z;       /* r = r * z mod n  */
      r = fmod(r1,n);
      }
    uno = fmod(x,2);     /* y = y div 2 */
    uno = x - uno;
    x = ceil(uno / 2);
    z1 = (z * z);
    z = fmod(z1,n);     /* z = z^2 mod n */
   }
*res = r;
return;
}
```

/* Generación de Claves Publica y Privada */

```
#include <math.h>
#include <stdlib.h>
#include <windows.h>

/* Procedimientos internos */
double mcd (double x, double fi);


/* Procedimientos exportables*/
void far pascal _export calcpub (double *c,double fi)
{
/* Condición:  1 < c < fi */
double x,m;

m = 0;
randomize();
x = random(fi);
/*(uno = fmod(x,2);
uno = x - uno;
x = ceil(uno/2); */

while (m != 1)
   {
     m = mcd(x,fi);
     x +=1;
   }
 x -=1;
*c = x;
return;
}


void far pascal _export calcpriv (double *d, double c,double fi)
/* d = exponente privado, c = exponente publico
   d = c^-1 mod fi
   es decir que (c * d) mod fi = 1 */
{
 double x1,x2,x3;
 double y1,y2,y3;
 double t1,t2,t3;
 double uno,q;

 x1 = 1;
 x2 = 0;
 x3 = fi;
 y1 = 0;
 y2 = 1;
 y3 = c;

 while (y3 > 1)
   {
    uno = fmod(x3,y3);
    uno = x3-uno;
```

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

```
    q = ceil(uno/y3); /*retorna la parte entera de la división */

    t1 = x1 - (q * y1);
    t2 = x2 - (q * y2);
    t3 = x3 - (q * y3);
    x1 = y1;
    x2 = y2;
    x3 = y3;
    y1 = t1;
    y2 = t2;
    y3 = t3;
   }
  *d = y2;
  return;
}


double mcd (double x,double fi)
{
double a,b,r;

a=fi;
b=x;
while (b != 0)
   {
   r= fmod (a,b);
   a=b;
   b=r;
   }
return(a);
}
```

/* Biblioteca de rutinas para encriptación y desencriptación con RSA y con Certificación de Claves */

```
#include <math.h>
#include <stdio.h>
#include <windows.h>

void expmod (double m,double y,double n,double *res);


void far pascal _export cifraRSACF (char *origen,char *destino,char
*certify,double exp,double n)
/* cada caracter de origen ^exp mod n */
{
double res,m;
int f, entero;
FILE *fo, *fd, *fc;


/*Apertura de archivos Origen, Certificado y Destino */
if ((fo=fopen(origen,"rb"))==NULL)
    printf("Error , NO se encontro el archivo %s !!!!\n",origen);
else
    if ((fc=fopen(certify,"rb"))==NULL)
      printf("Error , NO se encontro el archivo %s !!!!\n",certify);
    else
      fd=fopen(destino,"wb");

/* Encripta el mensaje */
while((f=fgetc(fo))!=EOF)
  {
   m = f;
   expmod(m,exp,n,&res);      /* Encripta exp = c */

   entero = res;
   fwrite(&entero,sizeof(entero),1,fd);
  }
f = 0;
fwrite(&f,sizeof(f),1,fd);
fclose(fo);

/* Encripta el certificado */
while((f=fgetc(fc))!=EOF)
  {
   m = f;
   expmod(m,exp,n,&res);      /* Encripta exp = c */

   entero = res;
   fwrite(&entero,sizeof(entero),1,fd);
  }
f = -1;
fwrite(&f,sizeof(f),1,fd);
fclose(fc);
fclose(fd);
}
```

```
void far pascal _export desenRSACF (char *origen,char *destino,char
*certify,double exp,double n)
/* cada caracter de origen ^exp mod n */
{
double res,m;
int entero;
FILE *fo, *fd, *fc;

/* Apertura de archivos */
if ((fo=fopen(origen,"rb"))==NULL)
     printf("Error , NO se encontro el archivo %s !!!!\n",origen);
else
    {
     fd=fopen(destino,"wb");
     fc=fopen(certify,"wb");
    }

/* Desencripta mensaje */
fread(&entero,sizeof(entero),1,fo);
while (entero!= 0)
   {
    m = entero;
    expmod(m,exp,n,&res);       /* Desencripta exp = d */

    fputc(res,fd);
    fread(&entero,sizeof(entero),1,fo);
   }
fclose(fd);


/* Desencripta certificado */
fread(&entero,sizeof(entero),1,fo);
while (entero!= -1)
   {
    m = entero;
    expmod(m,exp,n,&res);       /* Desencripta exp = d */

    fputc(res,fc);
    fread(&entero,sizeof(entero),1,fo);
   }
fclose(fc);
fclose(fo);
}
```

BIBLIOGRAFIA

[1] KONHEIM, Alan G. "Criptography: A primer". John Wiley & Sons, 1981.

[2] LUCCHESI, Cláudio Leonardo. "Introduçao á Criptografia Computacional". Campinas, Papirus, Unicamp, 1986.

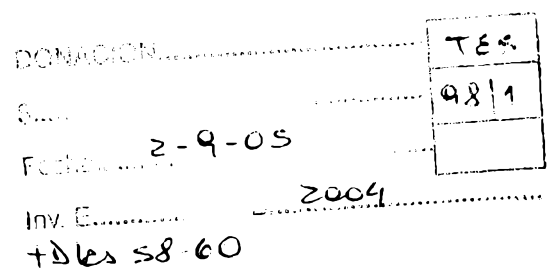
[3] RODRIGUEZ PRIETO, Amador. "Protección de la Información. Diseño de Criptosistemas Informáticos". Parainfo S.A., 1986.

[4] STALLINGS, Williams. "Network and Internetwork Security Principles and Practice". Prentice Hall, 1994.

BIBLIOTECA
FAC. DE INFORMÁTICA
U.N.L.P.

Otra información de interés

- Internet: www.rsa.com
- Internet: www.terisa.com
- Internet: www.scimitar.com
- Internet: www.math.uiuc.edu
- Internet: www.math.niu.edu



DONACION
Fecha 2-9-05
2004
Inv. E
+Dles 58.60
TES
98|1

AGRADECIMIENTOS

Agradezco al Ingeniero Horacio Villagarcía Wanza su guía y ayuda constante dedicada para el desarrollo de este trabajo.

Finalmente, agradezco profundamente el apoyo y el amor recibido en todo momento por mi esposo Joe, mis padres Juan Carlos y Mytta, mi hermana Julieta y toda mi familia, que me permitieron desarrollar este proyecto profesional tan anhelado.